ISO TC 184/SC4/WG3_N_789

Date: 23-Aug-99

Supersedes ISO TC 184/SC4/WG3 N729

ISO/[CD] [10303]-[216].[1]
Product data representation and exchange: Application Protocol: Ship moulded forms

COPYRIGHT NOTICE:
This ISO document is a working draft or committee draft and is copyright protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by Participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purposes of selling it should be addressed as shown below (via the ISO TC 184/SC4 Secretariat's member body) or to ISO's member body in the country of the requester.

Copyright Manager
ANSI
11 West 42nd Street
New York, New York 10036
USA
phone: +1-212-642-4900
fax: +1-212-398-0023

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

ABSTRACT: This part of ISO 10303 provides the application protocol for the exchange of ship moulded forms. The geometric shape and the hydrostatic properties of a ship moulded form are in scope.

KEYWORDS: ship moulded form, moulded form, design parameters, surface representation, wireframe representation, offset table representation, hydrostatics, intact stability

COMMENTS TO READER:
This document contains theCD version for the Ship Moulded Forms AP. The ARM model is based on the Ship Common Model SCM, which is used for the shipbuilding APs. For the Mapping Tables and the AIM the mapping guidelines and templates for shipbuilding APs developed in several AP 216/218 interpretation workshops are used. ARM and AIM of AP 216 and AP 218 are harmonised.

Project Leader: Frank Stolte

Address: Lloyd's Register of Shipping
Systems Development Group
TPDD
100 Leadenhall Street
London EC3A 3BP
Telephone: +44 171 423 1571
Telefacsimile: +44 171 423 2061
Electronic mail: Frank.Stolte@lr.org

Project Editor: Frank Stolte

Address: Lloyd's Register of Shipping
Systems Development Group
TPDD
100 Leadenhall Street
London EC3A 3BP
Telephone: +44 171 423 1571
Telefacsimile: +44 171 423 2061
Electronic mail: Frank.Stolte@lr.org

Contents

Page

1. Scope 1
2. Normative references 3
3. Terms, definitions and abbreviations 5
3.5 Abbreviations 7
4. Information requirements 9
4.1 Units of functionality 9
4.1.1 basic_geometry 12
4.1.2 configuration_management 12
4.1.3 date_time_resources 13
4.1.4 definitions 13
4.1.5 external_references 14
4.1.6 hydrostatics 14
4.1.7 items 15
4.1.8 location_concepts 15
4.1.9 measures_resources 16
4.1.10 offset_table_representations 17
4.1.11 organisation_resources 17
4.1.12 representations 18
4.1.13 shape 18
4.1.14 ship_design_parameter 22
4.1.15 ship_general_characteristics 23
4.1.16 ship_measures 24
4.1.17 ship_moulded_forms 24
4.1.18 surface_representations 25
4.1.19 wireframe_representations 25
4.2 Application objects 26
4.3 Application Assertions 123
5. Application interpreted model 136
5.1 Mapping table 136
5.2 AIM EXPRESS short listing 240
6. Conformance requirements 380

Annex A AIM EXPRESS expanded listing 382
Annex B AIM short names 494
Annex D Protocol Implementation Conformance Statement (PICS) proforma 496
Annex E Information object registration 497
Annex F Application activity model 498
F.1. Application activity model definitions and abbreviations 498
F.2. Application activity model diagrams 505
Annex G Application reference model 520
Annex H AIM EXPRESS-G 546
Annex J Computer interpretable listing 567
Annex K AP 216 Ship Moulded Forms usage guide 568
Annex L Technical discussions 569

Bibliography 575
Index 576

Figures

Figure 1 - Shipbuilding application protocols viii
Figure 2 - Ship moulded forms overview 2
Figure 3 - UoF used in AP 216 9
Figure 4- AP 216 structure 10
Figure 5 - Ship hull, rudder and propeller 26
Figure 6 - Ship hull with bulbous bow and thruster tunnels 27
Figure 7 – Ship hull with bulbous bow 29
Figure 8 – Bulkheads and decks 30
Figure 9 - Double bottom 30
Figure 10 – Outer and inner bottom with profiles 31
Figure 11 - Double ship hull 31
Figure 12 - Frames and decks 32
Figure 13 - Girders 32
Figure 14 - Screw propeller model 33
Figure 15 – Rudder and propeller 33
Figure 16 - Decks and superstructure 34
Figure 17 - Thruster 34
Figure 18 – Transom stern 35
Figure 19 – Transverse bulkheads 35
Figure 20 - Ship types 38
Figure 21 - Ship hull dimensions 55
Figure 22 – Midship tumble 59
Figure 23 - Keel dimensions 61
Figure 24 - Dimensions of the bulbous bow 62
Figure 25 - Propeller dimensions 64
Figure 26 – Controllable propeller blade 67
Figure 27 – Fixed propeller blade 67
Figure 28 – Ducted propeller 68
Figure 29 – Controllable screw propeller 69
Figure 30 - Thruster 69
Figure 31 - Vertical axis propeller 70
Figure 32 - Vertical axis propeller technology 71
Figure 33 - Propeller location 72
Figure 34 - Rudder dimensions 73
Figure 35 – Thruster propeller 76
Figure 36 – Bilge keel 77
Figure 37 - Shaft bossings 78
Figure 38 – Shaft strut 78
Figure 39 – Shaft struts 79
Figure 40 - Plate thickness 81
Figure 41 – Transverse meta- centre 83
Figure 42 - Form stability 86
Figure 43 - Wireframe 90
Figure 44 - Surface representation 92
Figure 45 - Ship points 92
Figure 46 - Ship curves 93
Figure 47 - Buttock lines 94
Figure 48 - Station lines 95
Figure 49 - Waterlines 95
Figure 50 - Global axis placements 98
Figure 51 – Axis 2 placement 3d 99
Figure 52 – Spacing position 102

Figure 53 – Key mappings for AP 216 138
Figure 54 - Node A0 moulded form lifecycle description 506
Figure 55 - Node A0 perform ship life cycle 507
Figure 56 - Node A1 specify ship 508
Figure 57 - Node A12 prepare bid 509
Figure 58 - Node A122 create preliminary design 510
Figure 59 - Node A1221 create preliminary hull form 511
Figure 60 - Node A12214 generate initial hull form definition 512
Figure 61 - Node A1223 estimate hydrodynamics and powering 513
Figure 62 - Node A12231 estimate resistance and powering 514
Figure 63 - Node A2 complete and approve ship design 515
Figure 64 - Node A22 finalise and approve hull form 516
Figure 65 - Node A23 finalise and approve hydrodynamics and powering 517
Figure 66 - Node A3 produce and inspect a ship 518
Figure 67 - Node A34 test ship 519
Figure 70 - ARM EXPRESS-G UoF Ship general characteristics diagram 1 522
Figure 71 - ARM EXPRESS-G UoF Ship general characteristics diagram 2 523
Figure 72 - ARM EXPRESS-G UoF Ship general characteristics diagram 3 524
Figure 73 - ARM EXPRESS-G UoF Ship general characteristics diagram 4 525
Figure 74 - ARM EXPRESS-G UoF Ship design parameter diagram 1 526
Figure 75 - ARM EXPRESS-G UoF Ship design parameter diagram 2 527
Figure 76 - ARM EXPRESS-G UoF Ship design parameter diagram 3 528
Figure 77 - ARM EXPRESS-G UoF Ship design parameter diagram 4 529
Figure 78 - ARM EXPRESS-G UoF Hydrostatics diagram 1 530
Figure 80 - ARM EXPRESS-G UoF Hydrostatics diagram 3 532
Figure 81 - ARM EXPRESS-G UoF Location concepts diagram 1 533
Figure 82 - ARM EXPRESS-G UoF Location concepts diagram 2 533
Figure 83 - ARM EXPRESS-G UoF Location concepts diagram 3 533
Figure 85 - ARM EXPRESS-G UoF Surface representations 534
Figure 86 - ARM EXPRESS-G UoF Wireframe representations 535
Figure 87 - ARM EXPRESS-G UoF Offset table representations 536
Figure 88 - ARM EXPRESS-G UoF Basic geometry diagram 1 536
Figure 89 - ARM EXPRESS-G UoF Basic geometry diagram 2 537
Figure 90 - ARM EXPRESS-G UoF Basic geometry diagram 3 537
Figure 91 - ARM EXPRESS-G UoF Items diagram 1 538
Figure 92 - ARM EXPRESS-G UoF Items diagram 2 538
Figure 93 - ARM EXPRESS-G UoF Items diagram 3 539
Figure 94 - ARM EXPRESS-G UoF Definitions 539
Figure 95 - ARM EXPRESS-G UoF Configuration management diagram 1 540
Figure 97 - ARM EXPRESS-G UoF Configuration management diagram 3 542
Figure 98 - ARM EXPRESS-G UoF Configuration management diagram 4 542
Figure 99 - ARM EXPRESS-G UoF External references diagram 1 543
Figure 100 - ARM EXPRESS-G UoF External references diagram 2 544
Figure 101 - ARM EXPRESS-G UoF Ship measures 545
Figure 102 – AIM EXPRESS-G diagram 1 of 21 546
Figure 103 – AIM EXPRESS-G diagram 2 of 21 547
Figure 104 – AIM EXPRESS-G diagram 3 of 21 548
Figure 105 – AIM EXPRESS-G diagram 4 of 21 549
Figure 106 – AIM EXPRESS-G diagram 5 of 21 550
Figure 107 – AIM EXPRESS-G diagram 6 of 21 551
Figure 108 – AIM EXPRESS-G diagram 7 of 21 552
Figure 109 – AIM EXPRESS-G diagram 8 of 21 553
Figure 110 – AIM EXPRESS-G diagram 9 of 21 554
Figure 112 – AIM EXPRESS-G diagram 11 of 21 556
Figure 114 – AIM EXPRESS-G diagram 13 of 21 558

Figure 115 – AIM EXPRESS-G diagram 14 of 21 559
Figure 116 – AIM EXPRESS-G diagram 15 of 21 560
Figure 117 – AIM EXPRESS-G diagram 16 of 21 561
Figure 118 – AIM EXPRESS-G diagram 17 of 21 562
Figure 119 – AIM EXPRESS-G diagram 18 of 21 563
Figure 120 – AIM EXPRESS-G diagram 19 of 21 564
Figure 121 – AIM EXPRESS-G diagram 20 of 21 565
Figure 122 – AIM EXPRESS-G diagram 21 of 21 566
Figure 123 - Ship Product Model 570
Figure 124 - Modelling framework 572
Figure 125 - Life cycle concept 573
Figure 126 - Redeclaration of attributes 573

Tables

Table 1 - Conformance classes 381

Foreword

ISO (the International Organization for Standardization)) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organisations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardisation.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

International Standard ISO 10303-216 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data*.

ISO 10303 consists of the following parts under the general title Industrial automation systems and integration — Product data representation and exchange:

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application interpreted constructs, application protocols, abstract test suites, implementation methods, and conformace testing. The series are described in ISO 10303-1.

A complete list of parts of ISO 10303 is available from the Internet:

`<http://www.nist.gov/sc4/editing/step/titles/>`

The structure of this International Standard is described in ISO10303-1. The numbering of the parts of the International Standard reflects its structure:

- Parts 11 to 13 specify the description methods,
- Parts 21 to 26 specify the implementation methods,
- Parts 31 to 35 specify the conformance testing methodology and framework,
- Parts 41 to 49 specify the integrated generic resources,
- Parts 101 to 106 specify the integrated application resources,
- Parts 201 to 234 specify the application protocols,
- Parts 301 to 331 specify the abstract test suites, and
- Parts 501 to 518 specify the application interpreted constructs.

Should further parts be published, they will follow the same numbering pattern.

Annexes A, B, C, D, E, F, G, H, and J form an integral part of this part of ISO 10303. Annexes K and L are for information only.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for archiving.

This part of ISO 10303 is a member of the application protocol series. This part of ISO 10303 specifies an application protocol (AP) for ship moulded forms and related hydrostatic properties.

The definition of ship moulded forms supports the geometrical representation of the ship hull, propellers, rudders, appendages and internal structures of the ship.

This part of ISO 10303 is one of a series of shipbuilding application protocols, that together aim to provide an integrated computer interpretable product model for ships.

The series of shipbuilding application protocols assumes that the ship product model can be divided into separate ship systems that each covers a key element of the ship for its whole life cycle. These key elements are: ship moulded forms, ship arrangements, ship distribution systems, ship structures, ship mechanical systems, ship outfit and furnishings, and ship mission systems. Each separate system is described by one or more different application protocols. The full series of shipbuilding application protocols is shown in Figure 1. Those aspects of the ship product model that are common to each shipbuilding application protocol are described consistently and identically in each application protocol.

Within the series of shipbuilding application protocols this part of ISO 10303 details the geometry of a ship moulded form.

A moulded form is the shape and a set of design parameters of different parts of the ship that does not include information on the thickness of the material from which it is constructed. A moulded form may describe a ship hull, propeller, rudder, appendage, deck or a ship structural element such as a bulkhead. A moulded form of particular interest is the ship hull, which is referred to as the hull moulded form. The hull moulded form will be exchanged between companies during the initial design and it is the basis of hydrostatic calculations.

All moulded forms are covered by this Part of ISO 10303, and a collection of moulded forms that describe the ship as a whole is termed a ship moulded form.

This application protocol satisfies an industrial need of reducing the time required for hull form design, performance prediction and ship structural design by facilitating the electronic exchange of hull moulded form geometry and hydrostatics between different companies. Also, it satisfies an industrial need for individual companies to integrate computer applications by providing an electronically accessible common view of hull and internal ship geometry for ship design and manufacture.

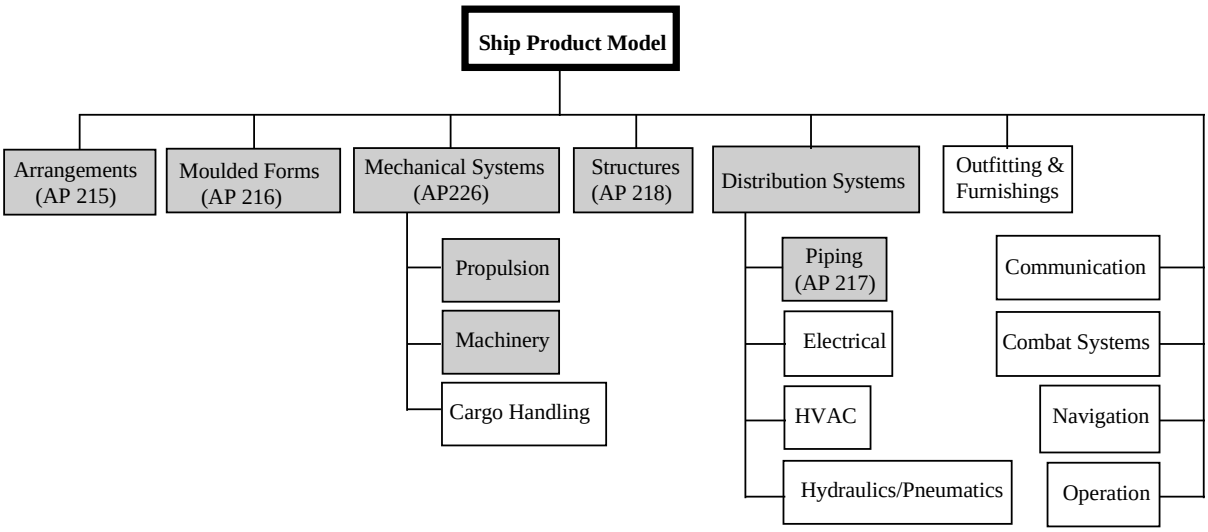


Figure 1 - Shipbuilding application protocols

The fundamental assumptions for ship moulded forms are:

— ship moulded form and each moulded form have a definition;

— definitions are approved and versioned but ship moulded form and moulded forms are not;

— a ship moulded form is associated with a ship and is composed of moulded forms;

— a moulded form provides a geometric representation for a ship.

This allows for the UoFs defined in this Part of ISO 10303 to be grouped under the following headings:

— moulded forms;

— product definition framework;

— support resources;

— utilities.

This is further described in chapter 4.1 and illustrated in Figure 3.

The representations that are described reflect the different input and output capabilities of CAD, CAM and other software systems in common usage of the shipbuilding process. The geometric representations supported are:

— offset table representation;

— wireframe representation;

— surface representation.

All geometric representations assume no knowledge of the thickness of the ship moulded form. All measurements are based on one set of units defined for the ship. All representations relate back to a ship moulded form definition, and so all geometric representations can be used to describe a hull moulded form or moulded forms of structures internal to the ship.

The hydrostatic properties are considered as a particular type of lifecycle definition for a moulded form. The hydrostatic properties are those of the intact hull that depend on the ship’s draught, such as

displacement, centre of buoyancy and centre of flotation. Damage stability is not covered since no information on ship compartmentation is described by this application protocol.

This application protocol defines the context, scope, and information requirements for the exchange of ship moulded form definitions, geometric representations, related hydrostatic properties, and specifies the integrated resources necessary to satisfy these requirements.

Application protocols provide the basis for developing implementations of ISO 10303 and abstract test suites for the conformance testing of AP implementations.

Clause 1 defines the scope of the application protocol and summarises the functionality and data covered by the AP. Clause 3 lists the words defined in this part of ISO 10303 and gives pointers to words defined elsewhere. An application activity modelthat is the basis for the definition of the scope, is provided in annex F. The information requirements of the application are specified in clause 4 using terminology appropriate to the application. A graphical representation of the information requirements, referred to as the application reference model, is given in annex G.

Resource constructs are interpreted to meet the information requirements. This interpretation produces the application interpreted model (AIM). This interpretation, given in 5.1, shows the correspondence between the information requirements and the AIM. The short listing of the AIM specifies the interface to the integrated resources and is given in 5.2. Note that the definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes, which are not imported into the AIM. The expanded listing given in Annex A contains the complete EXPRESS for the AIM without annotation. A graphical representation of the AIM is given in annex H. Additional requirements for specific implementation methods are given in annex C.

Industrial automation systems and integration — Product data representation and exchange — Part 216: Application Protocol : Ship Moulded Forms

1. Scope

This part of ISO 10303 specifies the scope and information requirements for the exchange of ship moulded form definitions, geometric representations, and related hydrostatic properties.

NOTE - The application activity model in annex F provides a graphical representation of the processes and information flows which are the basis for the definition of the scope of this part of ISO 10303.

An overview of the AP Ship Moulded Forms is given in Figure 2.

The following are within the scope of this part of ISO 10303:

— definition of moulded form geometry related to commercial and naval ships;

— definition of moulded form geometry of the preliminary design, detailed design and production stages of the ships life cycle;

— definition of moulded form geometry that describe the hull moulded form of the ship, including mono hull forms, multi-hull forms, the bulbous bow, transom stern, thruster tunnels, and additional appendages e.g., bilge keel, spray rails, shaft struts and shaft bossings that are part the final moulded form of the ship hull;

— definition of moulded form geometry that describe the moulded form of propellers and rudders

— definition of moulded geometry that describe the moulded form of decks including camber and sheer;

— definition of moulded geometry of internal ship compartment boundaries and the moulded form geometry of ship structural elements e.g., bulkheads, girders and profiles;

— definition of general characteristics such as main dimensions, shiptype, shipyard, shipowner, and classification data;

— definition of design parameter for the ship hull, bulbous bow, propeller, rudder, and appendages that are necessary in addition to describe the moulded form, and are required to calculate hydrostatic properties;

— definition of hydrostatic properties of the ship moulded form that depend on the ship's draught such as displacement, centre of buoyancy, centre of flotation, metacentric height, and cross curves of stability;

— definition of global and local co-ordinate systems and spacing tables used in naval architecture for position purposes;

— shape definition of ship moulded forms that use one of the following specified types of geometric representation:

a) offset table representation;

b) wireframe representation;

c) surface representation.

— geometric representations containing geometric elements used in naval architecture, such as waterlines and buttock lines;

— version control and approval of moulded forms and related hydrostatics.

The following are outside the scope of this part of ISO 10303:

— product definition data related to hull plating defined on the moulded form;

— product definition data related to ship compartmentation and ship arrangements;

— product definition data related to ship structures and ship assemblies;

— product definition data related to ship machinery and ship superstructures;

— mechanical systems and material aspects of propellers, rudders and control surfaces;

— product definition data from the decommissioning stage of the ship life cycle;

— hydromechanic properties of the ship such as powering, motion response and ship manoeuvring;

— damage stability properties of ships;

— ship longitudinal strength.

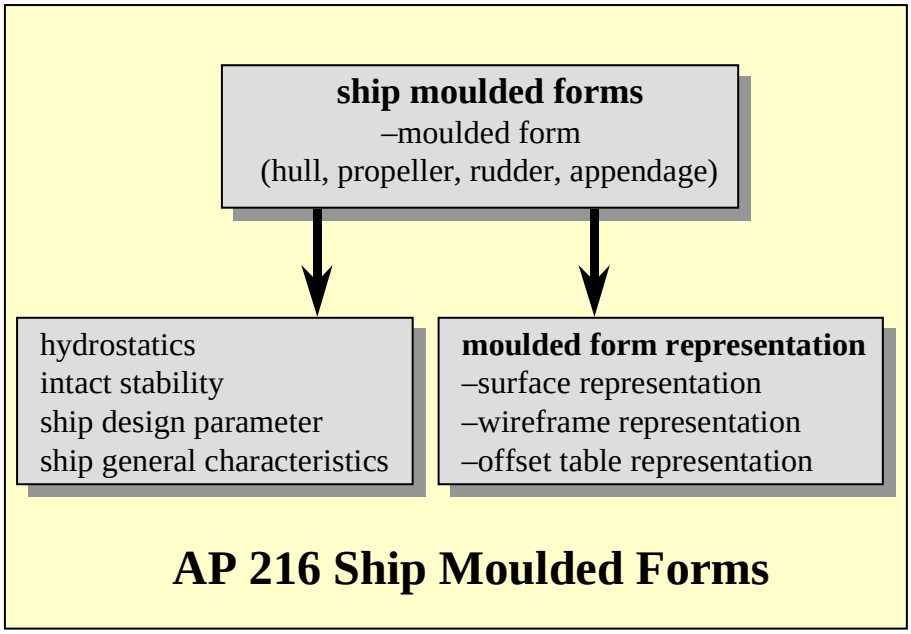


Figure 2 - Ship moulded forms overview

2. Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this part of ISO 10303. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 31:1994, *Quantities and Units*.

ISO 1000:1992, *SI units and recommendations for the use of their multiples and of certain other units.*

ISO 8824-1:1994. *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1) — Part 1: Specification of Basic notation*.

ISO 10303-1:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO/IEC 8824-1:1995, *Information Technology- Open Systems Interconnection- Abstract Syntax Notation One (ASN.1)-Part 1: Specification of Basic Notation.*

ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

ISO 10303-21:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.*

ISO 10303-31:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 31: Conformance testing methodology and framework: General concepts.*

ISO 10303-41:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resources: Fundamentals of product description and support.*

ISO 10303-42:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resources: Geometric and topological representation.*

ISO 10303-43:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resources: Representation structures.*

ISO 10303-44:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 44: Integrated generic resources: Product structure configuration.*

ISO 10303-45: *Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resources: Materials*

ISO 10303-46:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 46: Integrated generic resources*: Visual presentation.

ISO 10303-47: *Industrial automation systems and integration — Product data representation and exchange — Part 47: Integrated generic resources: Shape variation tolerances*.

ISO 10303-508:[1] *Industrial automation systems and integration — Product data representation and exchange — Part 502: Application interpreted constructs: Shell-based wireframe*

ISO 10303-510:[2] *Industrial automation systems and integration — Product data representation and exchange — Part 510: Application interpreted constructs: Non-manifold surface representation*

[1] To be published.
[2] To be published

3. Terms, definitions and abbreviations

3.1 Terms defined in ISO 10303-1

For the purposes of this part of ISO 10303, the following definitions defined in ISO 10303-1 apply.

— application;

— application activity model (AAM);

— application context;

— application interpreted model (AIM);

— application object;

— application protocol (AP);

— application reference model (ARM);

— computer aided design (CAD);

— conformance_class;

— conformance requirement;

— data;

— data exchange;

— implementation method;

— information;

— integrated resource;

— interpretation;

— PICS proforma;

— product;

— product data;

— unit of functionality (UoF).

3.2 Terms defined in ISO 10303-31

For the purposes of this part of ISO 10303, the following definitions defined in ISO 10303-31 apply.

— conformance testing;

— postprocessor;

— preprocessor.

3.3 Terms defined in ISO 10303-42

For the purposes of this part of ISO 10303, the following definitions defined in ISO 10303-42 apply.

— boundary;

— closed curve;

— connected;

— open curve;

— orientable.

3.4 Definitions for ISO 10303-216

For the purposes of this part of ISO 10303, the following definitions defined in ISO 10303-216 apply.

3.4.1

after perpendicular

the perpendicular at the intersection of the summer load waterline with the after side of the rudder post or the centreline of the rudder stock if there is no rudder post

3.4.2

amidships

a vertical plane arranged transversely in y/z direction of the global co-ordinate system at half-length between the after perpendicular and the forward perpendicular

3.4.3

camber

a planar curve that defines the curvature of a deck in the transverse direction

3.4.4

centreline

a line that is the result of the intersection of the centreplane and the hull moulded form. The centreline represents a special buttock

3.4.5

centreplane

a vertical plane arranged longitudinally in x/z direction of the global co-ordinate system that passes through the point at half of moulded breadth amidships

3.4.6

Classification Society

an organisation that enhances the safety of life and property at sea by providing rules, regulations and personnel for assessing and classifying ships during their life cycle

3.4.7

forward perpendicular

the perpendicular at the intersection of the summer load waterline with the fore side of the stem

3.4.8

pitching

a rotation of the ship about the transverse axis of the ship

3.4.9

rolling

a rotation of the ship about the longitudinal axis of the ship

3.4.10

sheer

a planar curve that defines the curvature of a deck in the longitudinal direction

3.4.11

stem

a planar curve that extends from the base of the ship to the uppermost deck at the centreline of the ship to form the bow of the ship

3.4.12

summer load waterline

the maximum draught at which a ship is permitted to sail during the summer months

3.4.13

weather deck

the upper-most watertight deck of the ship

3.4.14

yawing

a rotation of the ship about the vertical axis of the ship

3.5 Abbreviations

For the purposes of this Part of ISO 10303, the following abbreviations apply:

AAM	application activity model
AIM	application interpreted model
AP	application protocol

ARM	application reference model
BB	building block
CAD	computer aided design
CAM	computer aided manufacture
IACS	International Association of Classification Societies
IMO	International Maritime Organisation
PICS	protocol implementation conformance statement
SCM	Ship Common Model
SI	Système International
SOLAS	safety of life at sea
UoF	unit of functionality

4. Information requirements

This clause specifies the information required for the exchange of ship moulded form definitions, geometric representations and related hydrostatic properties.

The information requirements are specified as a set of units of functionality, application objects, and application assertions. These assertions pertain to individual application objects and to relationships between application objects. The information requirements are defined using the terminology of the subject area of this application protocol.

NOTE - A graphical representation of the information requirements is given in annex G. The information requirements correspond to those of the activities identified as being within the scope of this application protocol, in annex F. The mapping table specified in 5.1 shows how the integrated resources and application interpreted constructs are used to meet the information requirements of this application protocol.

4.1 Units of functionality

This sub-clause specifies the units of functionality for the ship moulded forms application protocol (see Figure 3 and Figure 4). This part of ISO 10303 specifies the following units of functionality:

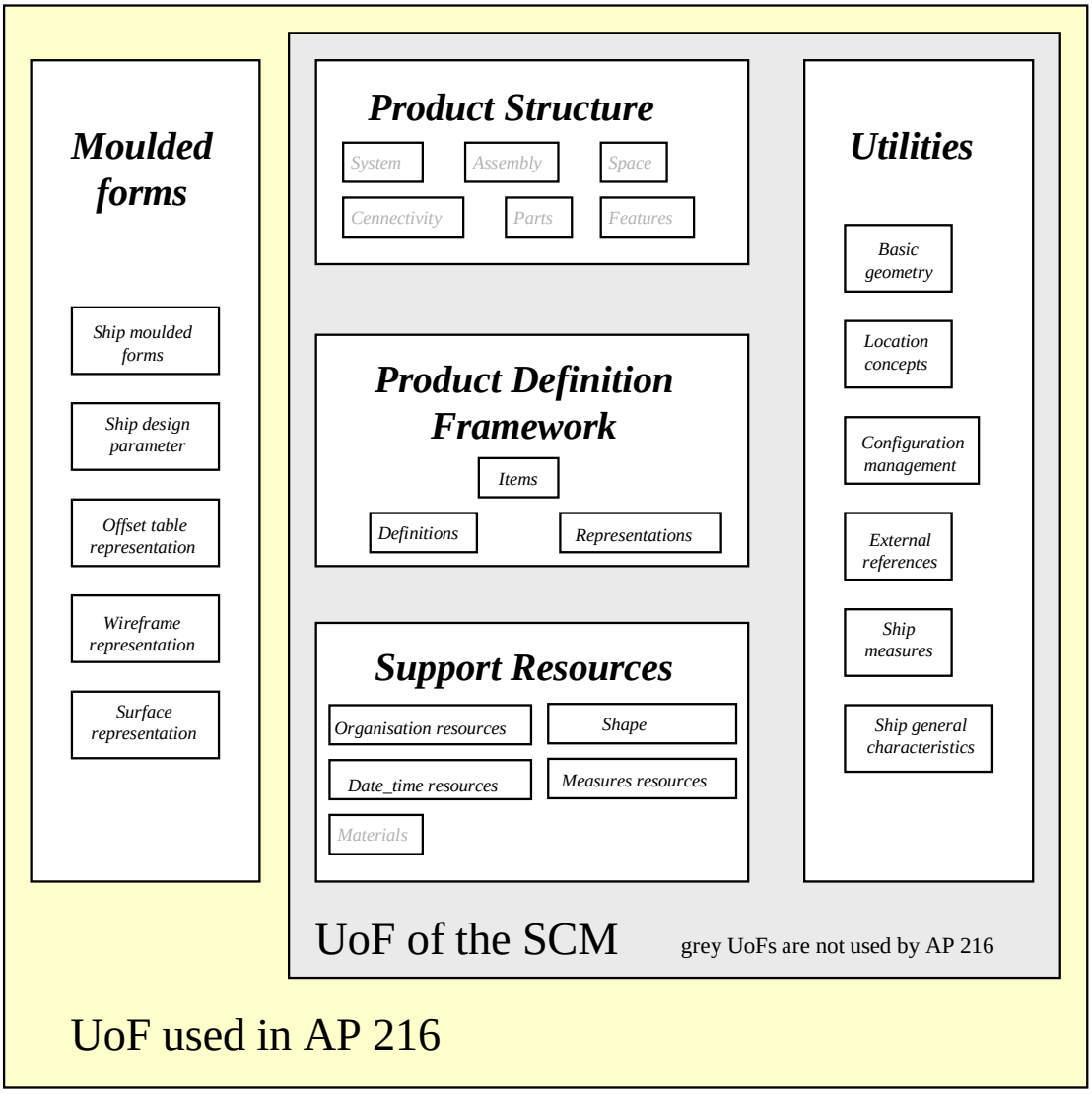


Figure 3 - UoF used in AP 216

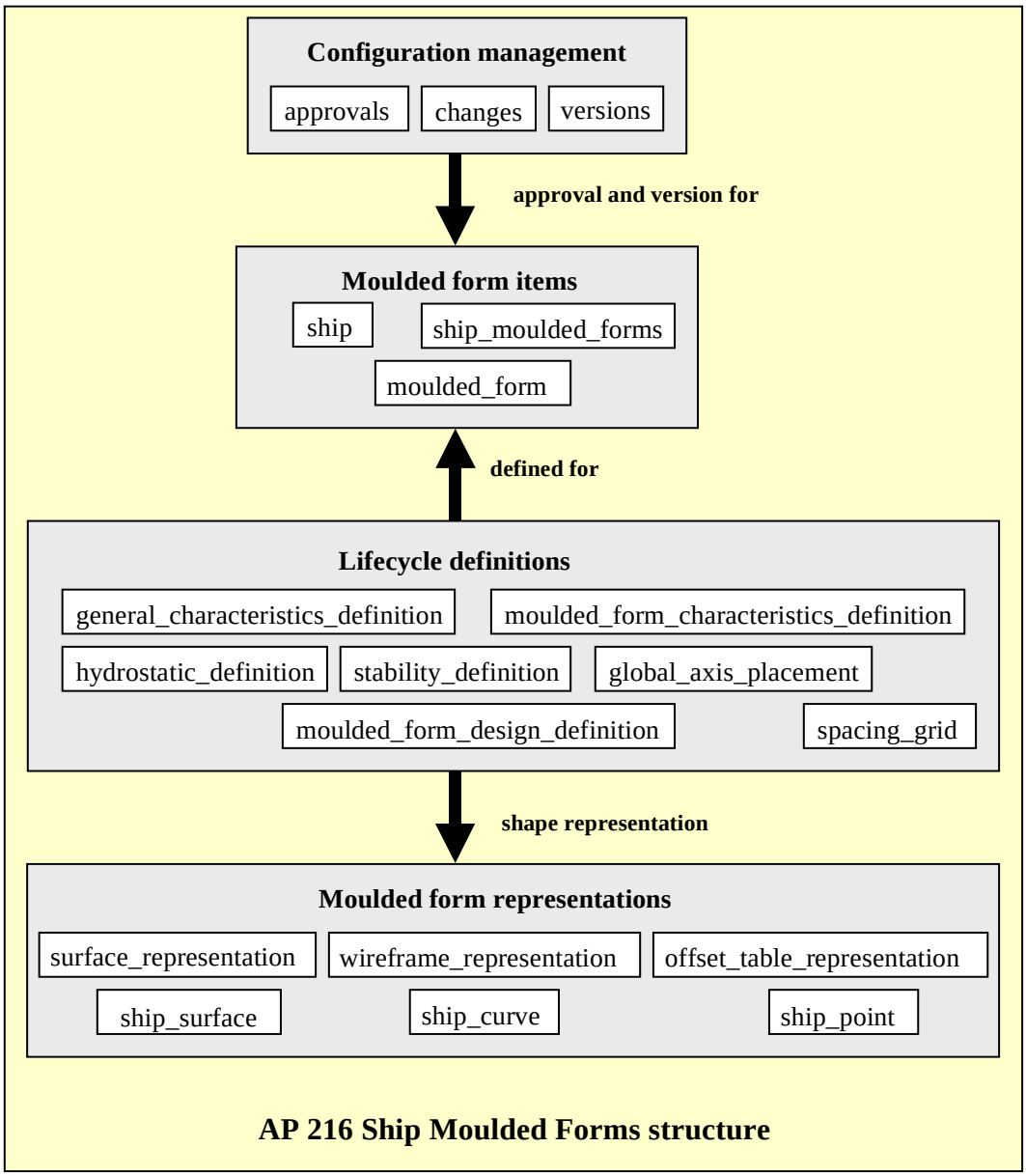


Figure 4- AP 216 structure

— basic_geometry;

— configuration_management;

— date_time_resources;

— definitions;

— external_references;

— hydrostatics;

— items;

— location_concepts;

— measures_resources;

— offset_table_representations;

— organisation_resources;

— representations;

— shape;

— ship_design_parameter;

— ship_general_characteristics;

— ship_measures;

— ship_moulded_forms;

— surface_representations;

— wireframe_representations.

The UoF's to be grouped under the following headings:

— moulded forms;

— product definition framework;

— product structure;

— support resources;

— utilities.

The group of UoF's labelled product definition framework provides the key part of the Ship Common Model and provides general concepts of how to relate things, how to define their properties and how to represent them.

The group of UoF's labelled product structure provides different levels of structuring a product mainly by space, by assembly or by system in a general manner.

The group of UoF's labelled moulded forms provides all specific information, which is required to define moulded forms and its properties in the AP 216.

The group of UoF's labelled support resources provides all STEP part 40ies resources used in shipbuilding AP's.

NOTE - All elements in this group will not be documented in clause 4.2 including the mapping tables as they are copies of the STEP part 40ies and are documented in the relevant STEP parts.

The group of UoF's labelled utilities provides the information for configuration management, location concepts, basic geometry and others, which is used by all shipbuilding AP's.

A general overview about the AP 216 data model is given in Figure 2. The units of functionality and a description of the functions that each UoF supports are given below and illustrated in Figure 3. The application objects included in the UoF's are defined in clause 4.2.

4.1.1 basic_geometry

This UoF provides the naval architecture nomenclature and categorisation given to points, curves and surfaces used in the representation of ship moulded forms.

The following application objects are used by the basic_geometry UoF:

- ship_curve;
- ship_curve_class_name;
- ship_curve_with_spacing_position;
- ship_point;
- ship_point_class_name;
- ship_surface;
- ship_surface_class_name;
- side_condition_type.

4.1.2 configuration_management

This UoF provides information required for configuration management to track versions, approval and changes of definitions. Versions provide the link to the definition, which is subject to versioning and provide concepts of how different versions are related to each other to provide a version history. Approvals provide the link to the definition, which is subject of approval and provide information about when, whom and to what level the approval took place. Changes provide the information about when and who changed which definition. Also, it describes the impact of a change in terms of whether or not other definitions are created, modified or deleted. The configuration_management UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the configuration_management UoF:

- approval_event;
- approval_history;
- approval_status;
- change;
- change_impact;
- change_plan;
- change_realization;
- change_request;
- change_state;
- check;
- definition_change_event;

— definition_creation;

— definition_deletion;

— definition_modification;

— envisaged_definition_creation;

— revision;

— revision_with_context;

— version_history;

— version_relationship.

4.1.3 date_time_resources

This UoF provides concepts to specify a date and time. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the date_time_resources UoF:

NOTE - The application objects defined in this UoF are identical to the objects defined in STEP part 41 date_time_schema and will not be documented in the 4.2 clause.

— calendar_date;

— date;

— date_and_time;

— local_time;

— ordinal_date;

— week_of_year_and_day_date.

4.1.4 definitions

This UoF provides the information for a general product definition based on product lifecycle or product structure and how definitions are related to each other.

The following application objects are used by the definitions UoF:

— definition;

— definition_relationship;

— design_definition;

— functional_definition;

— general_characteristics_definiton.

4.1.5 external_references

This UoF provides concepts for external references to something outside a given data exchange or data sharing context and provides information for the identification and reference of standards and documents defined outside ISO 10303. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the external_references UoF:

- any_address;
- any_author;
- document;
- document_definition;
- document_reference;
- document_referent;
- document_usage_constraint;
- external_instance_reference;
- external_reference;
- object_description;
- protocol_type;
- universal_resource_locator.

4.1.6 hydrostatics

This UoF provides the results of calculations of hydrostatic properties based on the extreme form associated to the displacement in still water. The extreme form describes the geometry, which is defined by the moulded form plus the thickness of the material.

The following application objects are used by the hydrostatics UoF:

- addition_of_moulded_form;
- displacement_operation;
- floating_position;
- hydrostatic_definition;
- hydrostatic_position_value;
- hydrostatic_property;
- hydrostatic_property_type;
- hydrostatic_property_value;

— hydrostatic_scalar_value;

— hydrostatic_table;

— stability_definition;

— stability_properties_for_one_floating_position;

— stability_property;

— stability_table;

— subtraction_of_moulded_form.

4.1.7 items

This UoF provides the information required for the definition of a general lifecycle independent structure of the product and relationships between its components. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the items UoF:

— definable_object;

— global_id;

— item;

— item_relationship;

— item_structure;

— ship;

— versionable_object.

4.1.8 location_concepts

This UoF provides different concepts of positioning for items of the ship. This includes the information required to define the ship's global co-ordinate system, local co-ordinate systems and spacing grids. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the location_concepts UoF:

— buttock_table;

— co_ordinate_system;

— frame_table;

— global_axis_placement;

— global_co_ordinate_system_orientation;

— local_co_ordinate_system;

— local_co_ordinate_system_with_position_reference;

— location_reference;

— longitudinal_position;

— longitudinal_table;

— spacing_position;

— spacing_position_with_offset;

— spacing_table;

— station_table;

— transversal_position;

— transversal_table;

— vertical_position;

— vertical_table;

— waterline_table.

4.1.9 measures_resources

This UoF provides the information for representing measures for physical quantities. The UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the measures_resources UoF:

NOTE - The application objects defined in this UoF are identical to the objects defined in STEP part 41 measures_schema and will not be documented in the 4.2 clause.

— area_measure;

— area_unit;

— derived_unit;

— derived_unit_element;

— dimensional_exponents;

— length_measure;

— length_unit;

— mass_measure,

— mass_unit;

— parameter_value;

— plane_angle_measure;

— positive_length_measure;

— ratio_measure;

— ratio_unit;

— si_prefix;

— si_unit;

— si_unit_name;

— unit;

— volume_measure;

— volume_unit.

4.1.10 offset_table_representations

This UoF provides the information required to geometrically representing a shape as a table of offset points. An offset table representation represents a shape entirely by a set of Ship_points. The Ship_points are listed in sections, normally these sections are 2D sections. In special cases there may be a need for using 3D sections.

The following application objects are used by the offset_table_representations UoF.

— offset_point_table_model;

— offset_table_shape_representation;

— offset_table_type;

— section_of_offset_point_table.

4.1.11 organisation_resources

This UoF provides the information for representing data about persons, organisations and their addresses. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the organisation_resources _resources UoF:

NOTE - The application objects defined in this UoF are identical to the objects defined in STEP part 41 person_organization_schema and will not be documented in the 4.2 clause.

— address;

— organization;

— organizational_address;

— person;

— person_and_organization;

— person_organization_select.

4.1.12 representations

This UoF provides the information required for generic representations associated with definitions and describes how representations are related to each other. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the representations UoF:

NOTE - The application objects defined in this UoF are identical to the objects defined in STEP part 41 representation_schema and will not be documented in the 4.2 clause.

— definitional_representation;

— functionally_defined_transformation;

— representation;

— representation_context;

— representation_item;

— representation_relationship;

— shape_representation.

4.1.13 shape

This UoF provides the information required for the definition of different types of surface and wireframe geometry including topological relations. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the shape UoF:

NOTE - The application objects defined in this UoF are identical to the objects defined in STEP part 42 geometric_model_schema, geometry_schema and topological_schema and will not be documented in the 4.2 clause.

— axis1_placement;

— axis2_placement;

— axis2_placement_2d;

— axis2_placement_3d;

— b_spline_curve;

— b_spline_curve_form;

— b_spline_curve_with_knots;

— b_spline_surface;

— b_spline_surface_form;

— b_spline_surface_with_knots;

— bezier_curve;

— bezier_surface;

— boundary_curve;

— bounded_curve;

— bounded_pcurve;

— bounded_surface;

— bounded_surface_curve;

— cartesian_point;

— cartesian_transformation_operator;

— cartesian_transformation_operator_2d;

— cartesian_transformation_operator_3d;

— circle;

— closed_shell;

— composite_curve;

— composite_curve_on_surface;

— composite_curve_segment;

— conic;

— conical_surface;

— connected_edge_set;

— connected_face_set;

— curve;

— curve_bounded_surface;

— curve_on_surface;

— curve_replica;

— cylindrical_surface;

— degenerate_pcurve;

— degenerate_toroidal_surface;

— dimension_count;

— direction;

— edge;

- edge_curve;
- edge_loop;
- elementary_surface;
- ellipse;
- evaluated_degenerate_pcurve;
- extent_enumeration;
- face;
- face_based_surface_model;
- face_bound;
- face_outer_bound;
- face_surface;
- geometric_representation_context;
- geometric_representation_item;
- hyperbola;
- intersection_curve;
- knot_type;
- line;
- list_of_reversible_topology_item;
- loop;
- offset_curve_2d;
- offset_curve_3d;
- offset_surface;
- open_path;
- open_shell;
- oriented_closed_shell;
- oriented_edge;
- oriented_face;
- oriented_open_shell;
- oriented_path;

— outer_boundary_curve;

— parabola;

— path;

— pcurve;

— pcurve_or_surface;

— placement;

— plane;

— point;

— point_on_curve;

— point_on_surface;

— point_replica;

— poly_loop;

— polyline;

— preferred_surface_curve_representation;

— quasi_uniform_curve;

— quasi_uniform_surface;

— rational_b_spline_curve;

— rational_b_spline_surface;

— rectangular_composite_surface;

— rectangular_trimmed_surface;

— reparametrised_composite_curve_segment;

— reversible_topology;

— reversible_topology_item;

— seam_curve;

— set_of_reversible_topology_item;

— shell;

— shell_based_wireframe_model;

— spherical_surface;

— subface;

- surface;
- surface_curve;
- surface_of_linear_extrusion;
- surface_of_revolution;
- surface_patch;
- surface_replica;
- swept_surface;
- topological_representation_item;
- toroidal_surface;
- transition_code;
- trimmed_curve;
- trimming_preference;
- trimming_select;
- uniform_curve;
- uniform_surface;
- vector;
- vector_or_direction;
- vertex;
- vertex_loop;
- vertex_point;
- vertex_shell;
- wire_shell.

4.1.14 ship_design_parameter

This UoF provides design parameter for the ship moulded form and the different moulded forms. This information contains dimensions and ratios important to the simple characterisation of the hull, deck, bow, propeller, rudder, appendage, thruster and keel details. The UoF also contains the information concerning the overall dimensions of the ship moulded form.

The following application objects are used by the ship_design_parameter UoF:

- appendage_moulded_form_design_parameter;
- appendage_type;

— bottom_moulded_form_design_parameter;

— bulb_moulded_form_design_parameter;

— deck_moulded_form_design_parameter;

— hull_moulded_form_design_parameter;

— kind_of_propeller_blade;

— location;

— midship_tumble;

— moulded_form_characteristics_definition;

— propeller_location;

— propeller_moulded_form_design_parameter;

— rudder_moulded_form_design_parameter;

— rudder_type;

— sense_of_rotation;

— ship_overall_dimensions;

— thruster_moulded_form_design_parameter.

4.1.15 ship_general_characteristics

This UoF provides the basic information that details the ship's dimensions and identification. The UoF contains dimension information for principal design parameters, parameters necessary for rule and regulation based calculations, as well as information about identification for ship owners, shipbuilders and classification societies. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the ship_general_characteristics UoF:

— class_and_statutory_designation;

— class_notation;

— class_parameters;

— commercialShipType;

— heavy_cargo;

— navalShipType;

— owner_designation;

— principal_characteristics;

— regulation;

- ship_designation;
- shipType;
- shipyard_designation;
- shipyard_role.

4.1.16 ship_measures

This UoF provides the definitions of additional measures for physical quantities, which are not supported by the measures_resources UoF. This UoF is common to all Shipbuilding related Application Protocols.

The following application objects are used by the ship_measures UoF:

- extended_conversion_based_unit;
- extended_measure_value;
- extended_measure_with_unit;
- speed_measure;
- speed_measure_with_unit;
- speed_unit.

4.1.17 ship_moulded_forms

This UoF provides the information required for the description of a moulded form, the composition of moulded forms to a ship moulded form, the representation of a moulded form and the handling of symmetry properties.

The following application objects are used by the ship_moulded_forms UoF:

- moulded_form;
- moulded_form_design_definition;
- moulded_form_functional_definition;
- moulded_form_functionality;
- moulded_form_relationship;
- moulded_form_representation_item;
- moulded_form_representation_relationship;
- moulded_form_shape_representation;
- planar_symmetry;
- rotational_symmetry;
- ship_moulded_form;

— ship_moulded_form_revision;

— symmetry.

4.1.18 surface_representations

This UoF specifies the information required to represent a moulded form as connected surfaces. The UoF combines topological and geometric information together to describe a surface model as a set of connected surface patches. This UoF describes simple geometry such as planes and straight lines, as well as complex geometry such as B-spline curves and surfaces. Additionally these may be associated with naval architecture terms. If the curves and surfaces are inter-connected then such information is stated explicitly by the use of topology.

The following application object is used by the surface_representations UoF:

— Surface_shape_representation.

4.1.19 wireframe_representations

This UoF specifies the information required to represent a moulded form as a wireframe. The UoF combines topological and geometric information together to describe a wireframe as a set of intersecting 2D or 3D curves. Both simple geometry such as straight lines and complex geometry such as B-spline curves are described by this UoF. Additionally these may be associated with naval architecture terms. Specific subtypes from STEP geometric resources are introduced to meet the shipbuilding requirements.

The following application objects are used by the wireframe_representations UoF:

— knot;

— ship_curve_segment;

— ship_moulded_form_wireframe_model;

— wireframe_shape_representation.

4.2 Application objects

This subclause specifies the application objects for the ship moulded forms application protocol. Each application object is an atomic element that embodies a unique application concept and contains attributes specifying the data elements of the object. The application objects and their definitions are given below.

NOTE - To improve the readability the application objects are ordered by UoFs. Figure 3 shows graphically the UoFs used in this part of ISO 10303.

I UoF ship_moulded_forms

4.2.1 Ship_moulded_form

A Ship_moulded_form is a type of Item_structure (see 4.2.85) and a type of Item (see 4.2.83) that groups all Moulded_forms together for the AP 216 data exchange. Typical examples for Moulded_forms are ship hull, propeller, rudder, appendages and others (see Figure 5).

Figure 5 - Ship hull, rudder and propeller

The data associated with a Ship_moulded_form are the following:

— items;

— relationships.

4.2.1.1 items

The items specify all Moulded_forms, which define the Ship_moulded_form. See 4.3.96 for the application assertion.

4.2.1.2 relationships

The relationships specify The Moulded_form_relationships, which form the Ship_moulded_form. See 4.3.97 for the application asssertion.

4.2.2 Moulded_form

A Moulded_form is a type of Item (see 4.2.83) that is part of the Ship_moulded_form. A Moulded_form defines the geometric shape of a constituent part of the ship. Certain types of Moulded_form may also be defined by a set of dimensions (see Figure 6).

NOTE - A Moulded_form does not include the thickness, or any other information, of the material from which it is constructed. There is a difference between a Moulded_form and a Ship_surface. The Moulded_form is a physical item, which is part of the Ship_moulded_form. A Ship's_surface instead is a representation of a Moulded_form. One or more Ship_surfaces represent a Moulded_form.

Figure 6 - Ship hull with bulbous bow and thruster tunnels

4.2.3 Moulded_form_relationship

A Moulded_form_relationship is a type of Item_relationship (see 4.2.84), which describes a relationship between two Moulded_forms. A collection of Moulded_form_relationships may describe a hierarchy of inter-related Moulded_forms that together provide a description of the Ship_moulded_form.

EXAMPLE - A typical example for a Moulded_form_relationship could be the relationship between the moulded_form of a ship hull and the related propeller.

The data associated with a Moulded_form_relationship are the following:

— item_1;

— item_2.

4.2.3.1 item_1

The item_1 specifies the Moulded_form that forms the basis of the Moulded_form_relationship. See 4.3.69 for the application assertion.

4.2.3.2 item_2

The item_2 specifies the Moulded_form that is to be associated with item_1. See 4.3.70 for the application assertion.

4.2.4 Moulded_form_design_definition

A Moulded_form_design_definition is a type of Design_definition (see 4.2.90) that specifies the definition of a Moulded_form.

The data associated with a Moulded_form_design_definition is the following:

— defined_for.

The defined_for specifies the Moulded_forms, which the Moulded_form_design_definition is defining. See 4.3.67 for the application assertion.

4.2.5 Moulded_form_functional_definition

The Moulded_form_functional_definition is a type of Functional_definition (see 4.2.91) that defines the function of a part of a ship, which is defined by the underlying Moulded_form.

The data associated with a Moulded_form_functional_definition are the following:

— defined_for;

— the_function.

4.2.5.1 defined_for

The defined_for specifies the Moulded_forms for which the Moulded_form_functional_definition is defined. See 4.3.68 for the application assertion.

4.2.5.2 the_function

The_function specifies the functionality of the Moulded_form for which the Moulded_form_functional_definition is defined.

The_function shall be one of the following:

— appendage,

— bulbous_bow,

— bulbous_stern,

— deck,

— double_bottom,

— double_ship_hull,

— frame,

— horizontal_girder,

— keel,

— longitudinal_bulkhead,

— longitudinal_girder,

— pressure_hull,

— propeller,

— rudder,

— ship_hull,

— superstructure,

— thruster,

— transom,

— transverse_bulkhead,

— user_defined.

4.2.5.2.1 appendage

appendage: any part, which is added to the ship hull to improve the properties of the ship (see Figure 36, Figure 37, Figure 38, and Figure 39).

EXAMPLE - Appendages can be bilge keel, shaft strut, shaft bossings, fins and others.

4.2.5.2.2 bulbous_bow

bulbous_bow: an underwater shape of the bow to smooth out bow wave and to reduce the resistance of the ship (see Figure 7).

Figure 7 – Ship hull with bulbous bow

4.2.5.2.3 bulbous_stern

bulbous_stern: an underwater shape of the stern to improve the flow to the propeller and to improve the efficiency of the propeller.

4.2.5.2.4 deck

deck: any horizontal dividing structure of the ship hull above the bottom or double bottom (see Figure 8).

Figure 8 – Bulkheads and decks

4.2.5.2.5 double_bottom

double_bottom: the internal bottom of a ship hull in case of double plating (see Figure 9).

Figure 9 - Double bottom

Figure 10 – Outer and inner bottom with profiles

4.2.5.2.6 double_ship_hull

double_ship_hull: the internal sides of the ship hull in case of double plating (see Figure 11).

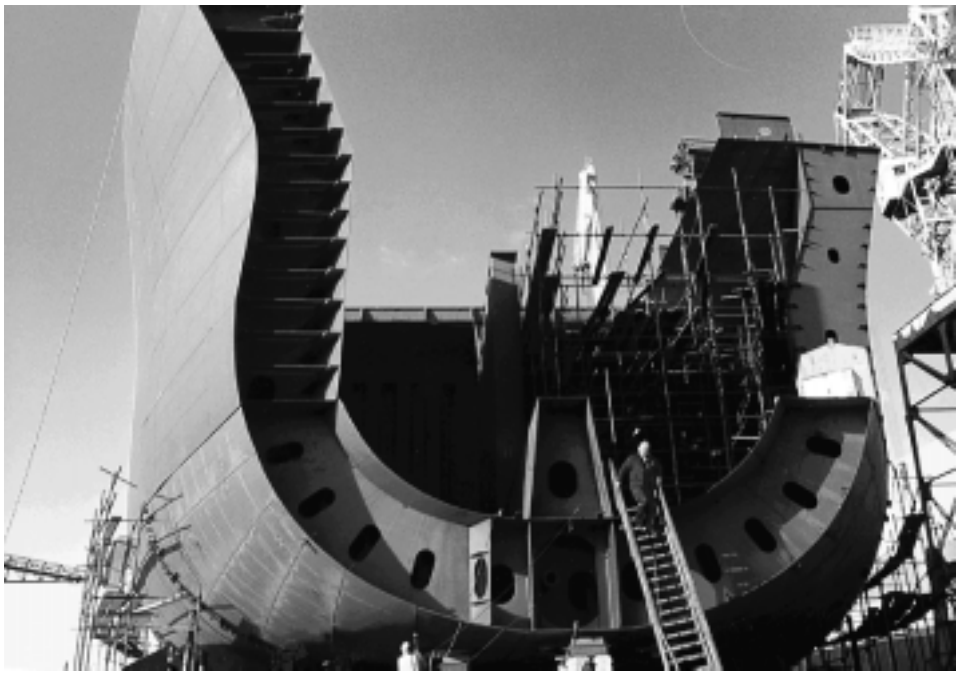


Figure 11 - Double ship hull

4.2.5.2.7 frame

frame: a vertical structural element arranged transversely in the ship that contributes significantly to the ship's strength (see Figure 12.

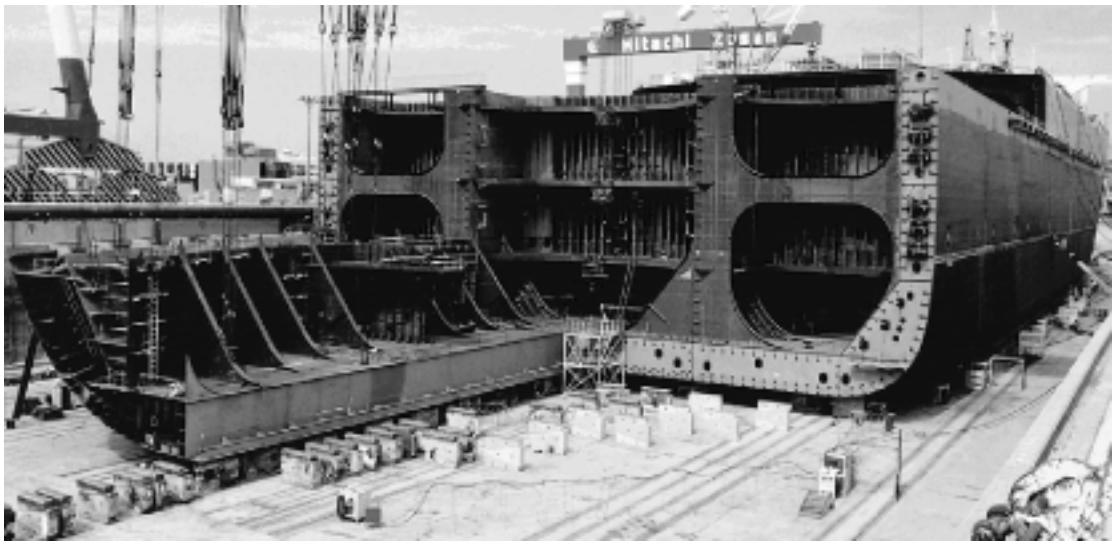


Figure 12 - Frames and decks

4.2.5.2.8 horizontal_girder

horizontal_girder: a longitudinal primary structural element that lies in the horizontal plane (see Figure 13).

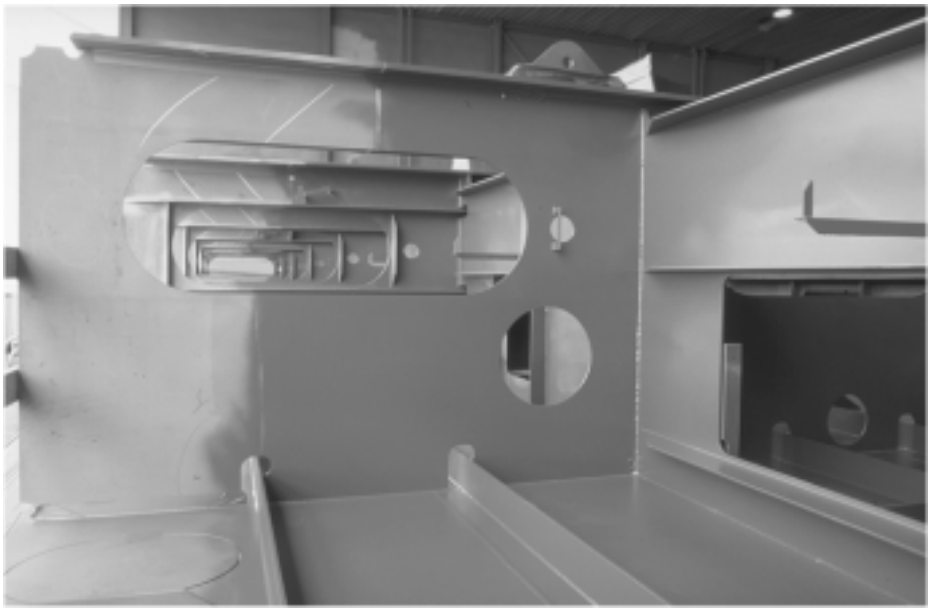


Figure 13 - Girders

4.2.5.2.9 keel

keel: the uninterrupted plating extending over the full length of the bottom from fore to aft of the ship. For sailing yachts the keel is specially shaped and includes additional weight to improve weight stability.

4.2.5.2.10 longitudinal_bulkhead

longitudinal_bulkhead: a vertical structural element arranged longitudinally in the ship that forms a compartment boundary and contributes significantly to the ship's sub-division and strength.

4.2.5.2.11 longitudinal_girder

longitudinal_girder: a longitudinal primary structural element that lies in the vertical plane (see Figure 13).

4.2.5.2.12 pressure_hull

pressure_hull: a special construction of a hull for underwater operation. The hull has to cope with underwater pressure and is used for submarines or any industrial underwater equipment.

4.2.5.2.13 propeller

propeller: converts the power of any engine into a force in the water for propelling or manoeuvring a ship (see Figure 14).

Figure 14 - Screw propeller model

4.2.5.2.14 rudder

rudder: main manoeuvring instrument for directional control of the ship (see Figure 15).

Figure 15 – Rudder and propeller

4.2.5.2.15 ship_hull

ship_hull: the main body of a ship for carrying goods or passengers or a body for stability and additional buoyancy (see Figure 11).

4.2.5.2.16 superstructure

superstructure: any deck structures above the upper deck e.g. the bridge (see Figure 16).

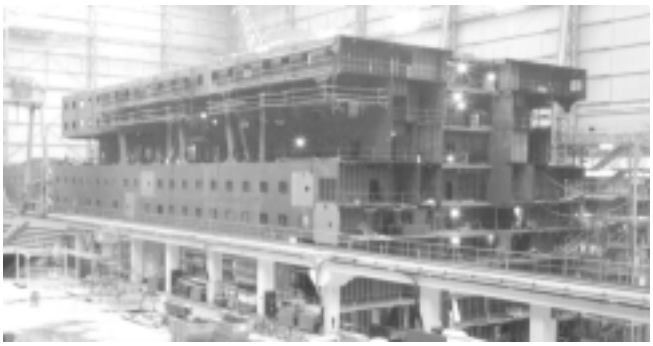


Figure 16 - Decks and superstructure

4.2.5.2.17 thruster

thruster: a transverse underwater tunnel, which runs across through the ship hull at the bow or stern of the ship. In the tunnel a propeller operates to enable better manoeuvring of the ship (see Figure 17).

Figure 17 - Thruster

4.2.5.2.18 transom

transom: after part of square sterned ship hull (see Figure 18).

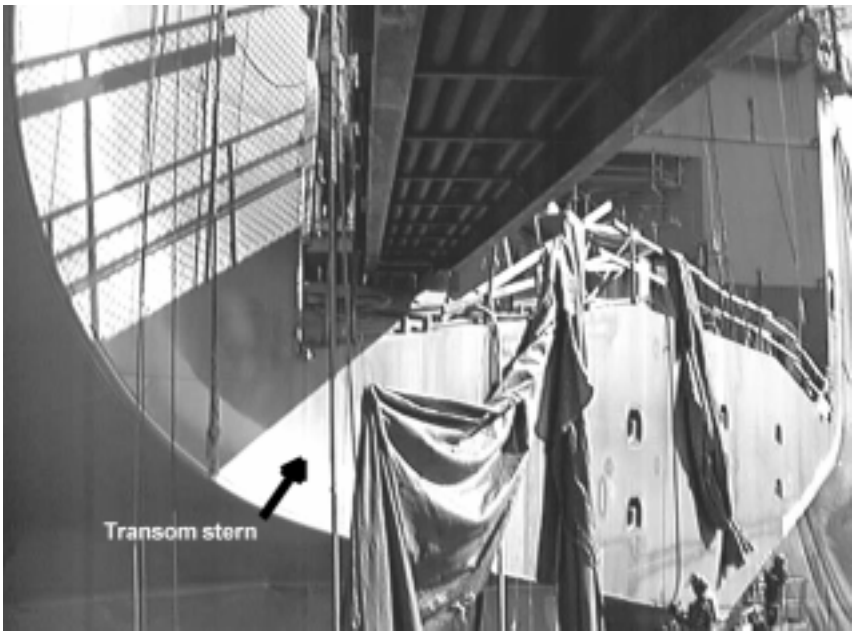


Figure 18 – Transom stern

4.2.5.2.19 transverse_bulkhead

transverse_bulkhead: a vertical primary structural element arranged transversely in the ship that forms a compartment boundary and contributes significantly to the ship's sub-division and strength (see Figure 19).

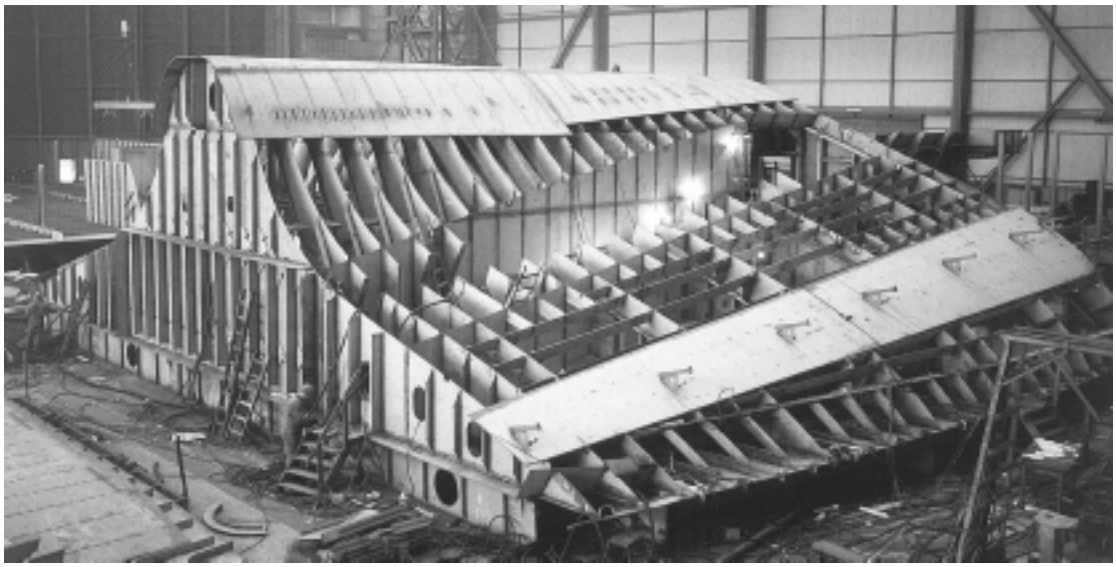


Figure 19 – Transverse bulkheads

4.2.5.2.20 user_defined

user_defined: a Moulded_form whose function is unknown or not covered by any of the above.

4.2.6 Ship_moulded_form_revision

A Ship_moulded_form_revision is a type of Revision_with_context (see 4.2.109) that details the revision of a Ship_moulded_form. It collects all used versions of Moulded_form_design_definitions, which are relevant for the definition of a specific ship moulded form.

The data associated with a Ship_moulded_form_revision are the following:

— context_of_revision;

— members.

4.2.6.1 context_of_revision

The context_of_revision specifies the Ship_moulded_form for which the Ship_moulded_form_revision is being defined. See 4.3.99 for the application assertion.

4.2.6.2 members

The members specify a collection of Moulded_form_design_definitions, which form the Ship_moulded_form_revision. See 4.3.98 for the application assertion.

4.2.7 Moulded_form_shape_representation

A Moulded_form_shape_representation is a type of Shape_representation that defines the shape of a Moulded_form using geometric models, topological and geometric elements. A Moulded_form_shape_representation is abstract in concept and does not detail the geometry but provides references to the topology and geometry that it may possess. The geometric constructs in a Moulded_form_shape_representation may reference a Ship_point, a Ship_curve or a Ship_surface. A Moulded_form_shape_representation may be an Offset_table_shape_representation (see 4.2.53), a Wireframe_shape_representation (see 4.2.56) or a Surface_shape_representation (see 4.2.60).

NOTE - Shape_representation is a STEP resource entity defined in ISO 10303-43.

The data associated with a Moulded_form_shape_representation are the following:

— moulded_form_representation_id;

— moulded_form_symmetry.

4.2.7.1 moulded_form_representation_id

The moulded_form_representation_id is optional and, if present, specifies the context specific label for the Moulded_form_shape_representation.

4.2.7.2 moulded_form_symmetry

The moulded_form_symmetry is optional and, if present, defines symmetry for the Moulded_form_shape_representation. Planar and rotational symmetry are possible for a Moulded_form. See 4.3.73 for the application assertion.

4.2.8 Moulded_form_representation_relationship

A Moulded_form_representation_relationship is a type of Representation_relationship that describes the relationship between two Moulded_form_shape_representations.

NOTE - Representation_relationship is a STEP resource entity defined in ISO 10303-43.

The data associated with a Moulded_form_representation_relationship are the following:

— id;

— rep_1;

— rep_2.

4.2.8.1 id

The id specifies the context specific identifier for the Moulded_form_representation_relationship.

4.2.8.2 rep_1

The rep_1 specifies the Moulded_form_shape_representation that forms the basis of the Moulded_form_representation_relationship. See 4.3.71 for the application assertion.

4.2.8.3 rep_2

The rep_2 specifies the Moulded_form_shape_representation that is to be associated with rep_1. See 4.3.72 for the application assertion.

4.2.9 Symmetry

Symmetry provides the symmetry information for any part of the ship. This can be either Planar_symmetry or Rotational_symmetry. A Symmetry may be a Planar_symmetry (see 4.2.10) or a Rotational_symmetry (see 4.2.11).

4.2.10 Planar_symmetry

A Planar_symmetry is a type of Symmetry (see 4.2.9) that provides the planar symmetry information for any part of the ship. The Planar_symmetry details the Symmetry_plane for this part of the ship.

The data associated with a Planar_symmetry is the following:

— the_symmetry_plane.

The_symmetry_plane specifies the plane of symmetry in the defining co-ordinate system of this part. The axis in axis2_placement_3d specifies the normal axis of the plane. If only one half of the part is defined by the geometry, then this should be the geometry describing the part to which the normal axis is pointing. See 4.3.79 for the application assertion.

4.2.11 Rotational_symmetry

A Rotational_symmetry is a type of Symmetry (see 4.2.9) that provides rotational symmetry information for any part of the ship. The rotational axis details the axis of rotation for this part of the ship.

The data associated with a Rotational_symmetry is the following:

— the_rotational_axis.

The_rotational_axis specifies the axis of rotation and is defined in the valid co-ordinate system of this part of the ship. See 4.3.88 for the application assertion.

4.2.12 Moulded_form_representation_item

A Moulded_form_representation_item is a type of Representation_item that represents representation items for a moulded form shape representation, which carry additional naval architectural meaning. A Moulded_form_representation_item may be a Ship_point (see 4.2.61), a Ship_curve (see 4.2.62) or a Ship_surface (see 4.2.64).

NOTE - Representation_item is a STEP resource entity defined in ISO 10303-43.

II UoF ship_general_characteristics

4.2.13 ShipType

A ShipType is a type of Functional_definition (see 4.2.91), which is a description of the function, purpose or mission for which a ship is designed. A ship may have multiple functions. In this case several Entities of this type shall be defined and assigned to an instance of Ship. It is almost impossible to provide a complete list of ship types. Therefore only the major types are named explicitly, other may be assigned via the description attribute (see Figure 20). A Ship_type may be a Carrier (see 4.2.14), a Working_ship (see 4.2.15), a Research_ship (see 4.2.16) or a Navy_ship (see 4.2.17).

Figure 20 - Ship types

The data associated with a ShipType are the following:

— defined_for;

— description.

4.2.13.1 defined_for

The defined_for specifies the ship for which the ship type is defined. See 4.3.104 for the application assertion.

4.2.13.2 description

The description attribute is used to assign a ship type, which is not covered in the list or needs further explanation.

4.2.14 Carrier

A Carrier is a type of Shiptype (see 4.2.13) and specifies a vessel, which transports goods or passengers.

The data associated with a Carrier is the following:

— has_type.

The has_type specifies the type of the carrier ship.

The has_type shall be one of the following:

— bulk_Carrier,

— car_Carrier,

— car_Ferry,

— chemical_Tanker,

— container_Carrier,

— cruise_liner,

— dry_cargo_Vessel,

— ferry,

— gas_carrier,

— highSpeedCraft,

— oil_Tanker,

— ore_Carrier,

— product_Tanker,

— reefer,

— roro_Vessel,

— user_defined.

4.2.14.1.1 Bulk_Carrier

Bulk_Carrier: a cargo ship constructed for transportation of Bulk.

4.2.14.1.2 Car_Carrier

Car_Carrier: a cargo ship constructed for the transport of cars from the manufacturer to end-user.

4.2.14.1.3 Car_Ferry

Car_Ferry: a cargo ship constructed for the transport of passengers, cars, coaches and lorries.

4.2.14.1.4 Chemical_Tanker

Chemical_Tanker: a cargo ship specially designed for transportation of chemicals.

4.2.14.1.5 Container_Carrier

Container_Carrier: a cargo ship constructed for the transport of containers.

4.2.14.1.6 Cruise_liner

Cruise_liner: ship constructed for the transport of passengers for pleasure.

4.2.14.1.7 Dry_cargo_Vessel

Dry_cargo_Vessel: a cargo ship constructed for the transport of any dry cargo, which is not packed but carried loose such as grain or coal.

4.2.14.1.8 Ferry

Ferry: a cargo ship constructed for the transport of passengers, cars, coaches, lorries and trains. The ferry could be designed for only one type of cargo e.g., passengers.

4.2.14.1.9 Gas_carrier

Gas_carrier: a cargo ship specially designed for transportation of gaseous products.

4.2.14.1.10 HighSpeedCraft

HighSpeedCraft: a cargo ship constructed for the high-speed transport of cargo.

4.2.14.1.11 Oil_Tanker

Oil_Tanker: a cargo ship specially designed for transportation of oil.

4.2.14.1.12 Ore_Carrier

Ore_Carrier: a cargo ship constructed for transportation of Ore.

4.2.14.1.13 Product_Tanker

Product_Tanker: a cargo ship specially designed for transportation of oil products.

4.2.14.1.14 Reefer

Reefer: a cargo ship constructed for the transport of any refrigerated cargo.

4.2.14.1.15 Roro_Vessel

Roro_Vessel: a cargo ship constructed for transportation of cargo, which can roll on board and roll off board.

4.2.14.1.16 user_defined

user_defined: can be any ship type which is not in the Enumeration list. Details may be found in the ship type description attribute.

4.2.15 Working_Ship

A Working_Ship is a type of Shiptype (see 4.2.13) and specifies a ship, which is constructed to perform specific working tasks.

The data associated with a Working_Ship is the following:

— has_type.

The has_type specifies the type of the working ship.

The has_type shall be one of the following:

— crane_Vessel,

— dredger,

— drilling_Vessel,

— fire_Fighter,

— fishing_Vessel,

— floating_Dock,

— floating_Hotel,

— FPGO,

— FPSO,

— ice_breaker,

— offshore_Supply_Vessel,

— oil_Production_Vessel,

— oil_Storage_Vessel,

— pilot_Boat,

— pipe_Laying_Vessel,

— pusher,

— shuttle_tanker,

— supply_Vessel,

— tug,

— user_defined.

4.2.15.1.1 Crane_Vessel

Crane_Vessel: a working ship constructed for lifting purposes.

4.2.15.1.2 Dredger

Dredger: a working ship constructed for dredging channels or harbour entrances.

4.2.15.1.3 Drilling_Vessel

Drilling_Vessel: a working ship constructed for drilling purposes.

4.2.15.1.4 Fire_Fighter

Fire_Fighter: a working ship constructed for fire fighting purposes.

4.2.15.1.5 Fishing_Vessel

Fishing_Vessel: a working ship constructed for fishing.

4.2.15.1.6 Floating_Dock

Floating_Dock: a working ship constructed for lifting ships for repair purposes.

4.2.15.1.7 Floating_Hotel

Floating_Hotel: a working ship constructed for hotel purposes.

4.2.15.1.8 FPGO

FPGO: a floating platform used for production and storage of gas.

4.2.15.1.9 FPSO

FPSO: a floating platform used for production and storage of oil.

4.2.15.1.10 Ice_breaker

Ice_breaker: a working ship constructed for breaking ice.

4.2.15.1.11 Offshore_Supply_Vessel

Offshore_Supply_Vessel: a working ship constructed for supplying offshore platforms.

4.2.15.1.12 Oil_Production_Vessel

Oil_Production_Vessel: a working ship constructed for production and storage of oil.

4.2.15.1.13 Oil_Storage_Vessel

Oil_Storage_Vessel: a working ship constructed for storage of oil.

4.2.15.1.14 Pilot_Boat

Pilot_Boat: a working ship constructed for carrying the pilot to the ship which he conducts in and out of the harbour.

4.2.15.1.15 Pipe_Laying_Vessel

Pipe_Laying_Vessel: a working ship constructed for pipe laying purposes.

4.2.15.1.16 Pusher

Pusher: a working ship constructed to push other unpropelled ships e.g. a barge.

4.2.15.1.17 Shuttle_tanker

Shuttle_tanker: a ship equipped for offshore oil loading. It may be used for transporting oil from a platform to shore.

4.2.15.1.18 Supply_Vessel

Supply_Vessel: a working ship constructed for supplying purposes e.g. offshore supply vessel.

4.2.15.1.19 Tug

Tug: a working ship constructed for towing purposes.

4.2.15.1.20 user_defined

user_defined: can be any ship type which is not in the Enumeration list. Details may be found in the ship type description attribute.

4.2.16 Research_Ship

A Research_Ship is a type of Shiptype (see 4.2.13) and specifies a ship, which is specially designed for research.

The data associated with a Research_Ship is the following:

— has_type.

The has_type specifies the type of the research ship.

The has_type shall be one of the following:

— user_defined.

4.2.16.1.1 user_defined

user_defined: can be any ship type which is not in the Enumeration list. Details may be found in the ship type description attribute.

4.2.17 Navy_Ship

A Navay_Ship is a type of Shiptype (see 4.2.13) and specifies a ship operating under military command.

The data associated with a Navy_Ship is the following:

— has_type.

The has_type specifies the type of the navy ship.

The has_type shall be one of the following:

— AIRCRAFT_CARRIER,

— AUXILIARY_OILER,

— CORVETTE,

— CRUISER,

— DESTROYER,

— FLEET_AUXILIARY_VESSEL,

— FRIGATE,

— LANDING_PLATFORM_DOCK,

— LANDING_PLATFORM_HELICOPTER,

— MINE_WARFARE_SHIP,

— PATROL_FORCE_VESSEL,

— SERVICE_CRAFT,

— SUBMARINE,

— user_defined.

4.2.17.1.1 AIRCRAFT_CARRIER

AIRCRAFT_CARRIER: the role of an aircraft carrier, apart from providing a Command, Control and Communications facility, is to deploy aircraft (either STOVL, STOBAR or CTOL) and or helicopters in sustained anti-submarine warfare operations and fighter protection, with full reconnaissance and strike capability.

4.2.17.1.2 AUXILIARY_OILER

AUXILIARY_OILER: a ship specially designed to replenish ships at sea with liquids during world-wide operations, with VERTREP services for the transfer of solids also being provided. The ship has a stable platform suitable for helicopters, including stowage and maintenance facilities plus emergency landing of other helicopters.

4.2.17.1.3 CORVETTE

CORVETTE: a ship specifically constructed for the purpose of performing the role of a small escort vessel to a task group. Corvettes are fitted primarily to fulfil an anti-submarine warfare role.

4.2.17.1.4 CRUISER

CRUISER: a ship typically used to screen carrier task forces. Cruisers (many with guided missiles or carrying a helicopter) provide anti-air warfare and anti-submarine capabilities. Cruisers also provide protection against anti-ship cruise missile threats at extended ranges, particularly in the presence of enemy electronic countermeasures.

4.2.17.1.5 DESTROYER

DESTROYER: a major surface combatant, which is typically used to conduct operations with strike, anti-submarine warfare and amphibious forces, and to perform screening and convoy duties. A destroyer may also be equipped with helicopters, providing an enhanced capability.

4.2.17.1.6 FLEET_AUXILIARY_VESSEL

FLEET_AUXILIARY_VESSEL: the primary task of a fleet auxiliary vessel (of which there are many types) is to supply warships at sea with fuel, food, stores and ammunition. An auxiliary also provides aviation platforms, amphibious support for the navy and marines and sea transport for army units.

4.2.17.1.7 FRIGATE

FRIGATE: a ship that can be generally described as a general purpose, ocean escort vessel. Operational requirements necessitate that frigates perform the duties of area defence ships, capable of defending a task group against modern day air threats. Frigates may also provide command facilities and accommodation. Secondary capabilities include anti-surface warfare, naval gunfire support and anti-submarine warfare

4.2.17.1.8 LANDING_PLATFORM_DOCK

LANDING_PLATFORM_DOCK: specifically designed to transport a large embarked military force (plus support equipment) across open oceans; support a flexible landing on hostile shores using onboard helicopters and landing craft, and co-ordinate the naval, air and land aspects of amphibious operations through Command, Control and Communications facilities.

4.2.17.1.9 LANDING_PLATFORM_HELICOPTER

LANDING_PLATFORM_HELICOPTER: an amphibious helicopter carrier designed to enable the rapid tactical deployment of airborne troops and equipment to spearhead amphibious operations ashore. It can also stand off a coast at strategic range in a deterrent operational role. Peacetime roles include troop and equipment transport, and humanitarian tasks such as disaster relief.

4.2.17.1.10 MINE_WARFARE_SHIP

MINE_WARFARE_SHIP: a mine countermeasures vessel (MCMV) which is specially constructed for the hunting, sweeping (by mechanical, acoustic or magnetic means) and clearance of mines in both inshore coastal and deep, exposed waters. Designed to have minimal magnetic signature, MCMV hulls are generally constructed of low magnetic steel, GRP or laminated wood.

4.2.17.1.11 PATROL_FORCE_VESSEL

PATROL_FORCE_VESSEL: the size of patrol force vessels can vary according to the roles for which they have been designed. Larger offshore patrol vessels (OPVs) are used for fire fighting, rescue or supply tasks, others are used as protection vessels, sometimes carrying a marine detachment and semi-rigid craft to act as a rapid response squadron. Some OPVs have ice-strengthened hulls and a helicopter landing deck for operation as survey vessels in the Arctic or Antarctic regions. The size, manoeuvrability and other operational characteristics of smaller coastal patrol craft means that they are ideally suited for patrol, search and rescue duties in coastal areas, or ports, harbours and other restricted waters.

4.2.17.1.12 SERVICE_CRAFT

SERVICE_CRAFT: primarily small vessels that have been specially designed to provide services to a fleet in harbours and ports. These include tugs, tenders, barges, patrol craft, recovery vessels, floating docks, etc. Other service craft providing support are larger ocean-going vessels, such as transports, survey and research ships, repair vessels, cargo ships, hospital ships, etc.

4.2.17.1.13 SUBMARINE

SUBMARINE: a high-capability, ocean-going vessel designed typically to perform both anti-submarine and anti-surface warship tasks. To perform these functions, generally a submarine carries heavyweight torpedoes, underwater-to-surface guided weapons or submarine-laid mines. Submarines are able to operate submerged in shallow waters as well as the open ocean, whilst also being capable of operation in waters from the Tropics to the Arctic.

4.2.17.1.14 user_defined

user_defined: can be any ship type which is not in the Enumeration list. Details may be found in the ship type description attribute.

4.2.18 Class_parameters

Class_parameters are a type of General_characteristics_definition (see 4.2.92) that specify the length and speed of the ship in accordance with Classification Society rules and statutory regulations.

The data associated with a Class_parameters are the following:

— block_coefficient_class;

— design_speed_ahead;

— design_speed_astern;

— froude_number;

— length_class;

— length_solas;

— scantlings_draught.

4.2.18.1 block_coefficient_class

The block_coefficient_class specifies the ratio of the moulded displacement volume to the volume of a block that has its length equal to the length_class, its breadth equal to the moulded_breadth and its depth equal to the scantlings_draught.

4.2.18.2 design_speed_ahead

The design_speed_ahead specifies the forward speed at which the ship is designed to operate.

4.2.18.3 design_speed_astern

The design_speed_astern specifies the reverse speed at which the ship is designed to operate.

4.2.18.4 froude_number

The froude_number specifies a parameter of significance to the resistance and propulsion calculations of a ship. The Froude number is a nondimensional parameter which is defined by dividing the ship speed in meter per second by the square root of the product of the length of the ship in meter and the gravitational acceleration in meter per square second. In the DERIVE clause the design_speed_ahead for the ship speed and the length_class for the length of the ship are used.

NOTE - Algebraically the Froude number could be given as $\mathbf{F = V/\sqrt{(gL)}}$ where V is the design_speed_ahead, L is the length_class and g is the gravitational constant.

4.2.18.5 length_class

The length_class specifies a length measurement for the ship that is defined in Classification Society Rules.

4.2.18.6 length_solas

The length_solas specifies a length measurement for the ship measured in accordance with the International Convention on the Safety of Life at Sea.

4.2.18.7 scantlings_draught

The scantlings_draught specifies the summer load draught used by the Classification Society in its calculations for structural integrity and strength.

4.2.19 Principal_characteristics

A Principal_characteristics is a type of General_characterisics_definition (see 4.2.92) that specify the main shape parameters of the hull moulded form. Principal_characteristics also include data that is required in subsequent iterations of the hull development process when one is considering hydrostatics.

The data associated with a Principal_characteristics are the following:

— block_coefficient;

— design_deadweight;

— design_draught;

— length_between_perpendiculars;

— max_draught_at_AP;

— max_draught_at_FP;

— min_draught_at_AP;

— min_draught_at_FP;

— moulded_breadth;

— moulded_depth.

4.2.19.1 block_coefficient

The block_coefficient is optional and if present, specifies the ratio of the moulded displacement volume to the volume of a block that has its length equal to the length_between_perpendiculars, its breadth equal to the maximum immersed moulded_breadth and its depth equal to the design_draught. The block_coefficient should be defined only for mono hull ships.

4.2.19.2 design_deadweight

The design_deadweight specifies the weight of the ship representing the weight of cargo, bunker fuel, water, passengers, crew and consumables that a ship can carry when loaded to the summer load line.

4.2.19.3 design_draught

The design_draught specifies the draught to which the ship has been designed to operate.

4.2.19.4 length_between_perpendiculars

The length_between_perpendiculars specifies the length measured from the after perpendicular to the forward perpendicular of the ship.

4.2.19.5 max_draught_at_AP

The max_draught_at_AP specifies the maximum possible draught at the Aft Perpendicular during the operation of the ship. The max_draught_at_AP is used for hull cross section approval for ice class notation.

4.2.19.6 max_draught_at_FP

The max_draught_at_FP specifies the maximum possible draught at the Forward Perpendicular during the operation of the ship. The max_draught_at_FP is used for hull cross section approval for ice class notation.

4.2.19.7 min_draught_at_AP

The min_draught_at_AP specifies the minimum possible draught at the Aft Perpendicular during the operation of the ship. The min_draught_at_AP is used for hull cross section approval for ice class notation.

4.2.19.8 min_draught_at_FP

The min_draught_at_FP specifies the minimum possible draught at the Forward Perpendicular during the operation of the ship. The min_draught_at_FP is used for hull cross section approval for ice class notation.

4.2.19.9 moulded_breadth

The moulded_breadth specifies the maximum breath of the ship amidships and at the design_draught.

4.2.19.10 moulded_depth

The moulded_depth specifies the vertical distance above the baseline to the uppermost deck where the deck joins the side of the ship measured amidships.

4.2.20 Class_and_statutory_designation

Class_and_statutory_designation is a type of General_characteristics_definition (see 4.2.92) that specifies the identification given to the ship by the classification society for the purpose of design, manufacture and in service approval.

The data associated with a Class_and_statutory_designation are the following:

— class_number;

— the_class;

— the_statutory.

4.2.20.1 class_number

The class_number specifies a classification society specific identifier to a ship.

4.2.20.2 the_class

The the_class specifies the applicable Class_notation with information about the ship type and the cargo. See 4.3.16 for the application assertion.

4.2.20.3 the_statutory

The the_statutory specifies the set of national and international regulations and standards with which the ship is intended to comply. See 4.3.17 for the application assertion.

4.2.21 Class_notation

Class_notation specifies the notations given to the ships hull and machinery by the classification society as a result of its approval activities during the design, manufacture and in service maintenance of the ship.

The data associated with a Class_notation are the following:

— approval_required_for_heavy_cargo;

— approval_required_for_oil_cargo;

— approval_required_loading_unloading_aground;

— approval_required_loading_unloading_grabs;

— class_notations_hull;

— class_notations_machinery;

— class_society;

— ice_class_notation;

— service_area;

— service_factor.

4.2.21.1 approval_required_for_heavy_cargo

The approval_required_for_heavy_cargo is optional and if present, specifies a flag indicating whether or not approval for special strengthening for heavy cargoes is necessary. These notations are valid for bulk carriers to indicate the distribution of loads across the cargo holds.

The approval_required_for_heavy_cargo shall be one of the following:

— HC,

— HC_E,

— HC_EA.

4.2.21.1.1 HC

HC: strengthened for heavy cargo. Heavy bulk cargo may be unevenly distributed among the cargo holds.

4.2.21.1.2 HC_E

HC_E: strengthened for heavy cargo. In addition to HC a non-homogeneous loading condition with empty holds on full draught is approved. The approved combination of empty holds is added to the notation (Holds 2, 3, 5 empty).

4.2.21.1.3 HC_EA

HC_EA: strengthened for heavy cargo. Any cargo hold may be empty at full draught. The approved combinations of empty holds are added to the notation (Holds 2, 3, 5 empty or ... empty).

4.2.21.2 approval_required_for_oil_cargo

The approval_required_for_oil_cargo specifies a flag indicating whether or not approval is required for the carriage of oil cargoes.

4.2.21.3 approval_required_loading_unloading_aground

The approval_required_loading_unloading_aground specifies a flag indicating whether or not approval for loading and unloading aground is necessary.

4.2.21.4 approval_required_loading_unloading_grabs

The approval_required_loading_unloading_grabs specifies a flag indicating whether or not approval for loading and unloading using grabs is necessary.

4.2.21.5 class_notations_hull

The class_notations_hull specifies the notation given to the ship hull by the classification society as a result of its approval activities done on the hull.

4.2.21.6 class_notations_machinery

The class_notations_machinery specifies the notation given to the ships machinery by the classification society as a result of its approval activities done on the machinery.

4.2.21.7 class_society

The class_society specifies the name and organisational details of the classification society whose rules and regulations are being used to assess the ship. See 4.3.18 for the application assertion.

4.2.21.8 ice_class_notation

The ice_class_notation is optional and if present, specifies the type of class notation given to the ship indicating the ice conditions in which the ship has been approved to operate.

4.2.21.9 service_area

The service_area describes the area or route in which the ship operates. This may include information about waterway, wave, weather and wind conditions.

4.2.21.10 service_factor

The service_factor is optional and if present, is related to the service area of the ship and the waves that occur in that area. The service_factor should be in the range of 0.5 to 1.

4.2.22 Regulation

A Regulation specifies the set of international and national regulations as well as standards, which apply to the ship.

The data associated with a Regulation are the following:

— international_regulations;

— national_regulations;

— standards.

4.2.22.1 international_regulations

The international_regulations specify all relevant international regulations, which apply to the ship. See 4.3.83 for the application assertion.

4.2.22.2 national_regulations

The national_regulations specify all relevant national regulations, which apply to the ship. See 4.3.84 for the application assertion.

4.2.22.3 standards

The standards specify all relevant standards, which apply to the ship. See 4.3.85 for the application assertion.

4.2.23 Ship_designation

A Ship_designation is a type of General_characteristics_definition (see 4.2.92), which specifies the identification given to the ship in order that it can be categorised by any shipping related organisation.

The data associated with a Ship_designation are the following:

— call_sign;

— flag_state;

— port_of_registration;

— ship_identification;

— ship_name;

— ship_type_description.

4.2.23.1 call_sign

The call_sign is a unique, lifecycle identifier assigned to the ship by the flag_state for radio communication.

4.2.23.2 flag_state

The flag_state specifies the national authority with which the ship is registered.

4.2.23.3 port_of_registration

The port_of_registration specifies the national homeport of the ship. The port_of_registration lies within the jurisdiction of the flag_state.

4.2.23.4 ship_identification

The ship_identification is a general identifier unique to the ship assigned during the classification process.

4.2.23.5 ship_name

The ship_name specifies the name of the ship assigned by the owner.

4.2.23.6 ship_type_description

The ship_type_description gives more details about the function of the ship and additional information about the cargo carried. If the ship_type is user_defined, then the ship_type_description delivers the information for the type of the ship.

4.2.24 Shipyard_designation

A Shipyard_designation is a type of General_characteristics_definition (see 4.2.92) that specifies the identification given to the ship by the shipbuilder.

The data associated with a Shipyard_designation are the following:

— role;

— shipyard;

— shipyard_new_building_id;

— shipyard_project_name.

4.2.24.1 role

The role specifies the contractual obligation the shipyard has in relation to the ship.

The role shall be one of the following:

— prime,

— prime_build,

— prime_design,

— prime_repair,

— subcontractor.

4.2.24.1.1 prime

prime: the prime contractor for the ship

4.2.24.1.2 prime_build

prime_build: the prime contractor with contract responsibilities for manufacture of the ship

4.2.24.1.3 prime_design

prime_design: the prime contractor with contract responsibility for the design of the ship

4.2.24.1.4 prime_repair

prime_repair: the prime contractor with contract responsibilities for repair of the ship

4.2.24.1.5 subcontractor

subcontractor: a subcontractor for the ship

4.2.24.2 shipyard

The shipyard specifies the name and organisational details of the shipyard. See 4.3.105 for the application assertion.

4.2.24.3 shipyard_new_building_id

The shipyard_new_building_id is optional and if present, specifies an identifier for the ship that is assigned by the shipyard after an order has been confirmed.

4.2.24.4 shipyard_project_name

The shipyard_project_name specifies an identifier for the ship that is assigned by the shipyard on receipt of an order, or tender, for a new ship

4.2.25 Owner_designation

An Owner_designation is a type of General_characteristics_definition (see 4.2.92) that specifies the organisations that order, own and manage the ship.

The data associated with an Owner_designation are the following:

— managing_company;

— ordering_company;

— owning_company.

4.2.25.1 managing_company

The managing_company specifies the organisation that is responsible for managing and operating the ship. See 4.3.76 for the application assertion.

4.2.25.2 ordering_company

The ordering_company specifies the organisation that ordered the ship at a shipyard. See 4.3.77 for the application assertion.

4.2.25.3 owning_company

The owning_company specifies the organisation that legally owns the ship. See 4.3.78 for the application assertion.

III UoF ship_design_parameter

4.2.26 Ship_overall_dimensions

Ship_overall_dimensions are a type of definition (see 4.2.88) that specify the maximum limits of the Ship_moulded_form (see Figure 21).

The data associated with a Ship_overall_dimensions are the following:

— defined_for;

— overall_breadth;

— overall_depth;

— overall_length;

— stem_overhang;

— stern_overhang.

4.2.26.1 defined_for

The defined_for identifies the Ship_moulded_form for which the Overall_dimensions are defined for. See 4.3.101 for the application assertion.

4.2.26.2 overall_breadth

The overall_breadth specifies the maximum breadth of the Ship_moulded_form.

4.2.26.3 overall_depth

The overall_depth specifies the maximum vertical distance from the baseline to the uppermost deck of the Ship_moulded_form.

4.2.26.4 overall_length

The overall_length specifies the length between the extreme forward and aft ends of the Ship_moulded_form.

4.2.26.5 stem_overhang

The stem_overhang specifies the length from the forward perpendicular to the extreme forward end of the Ship_moulded_form.

4.2.26.6 stern_overhang

The stern_overhang specifies the length from the extreme aft end of the Ship_moulded_form to the after perpendicular.

4.2.27 Moulded_form_characteristics_definition

The Moulded_form_characteristics_definition is a type of Definition (see 4.2.88) that specifies design parameter for the different Moulded_forms. This information contains dimensions and ratios important to the simple characterisation of the hull, bow, propeller, rudder, appendage, thruster and bottom details. A Moulded_form_characteristics_definition may be a Hull_moulded_form_design_parameter (see 4.2.28), a Deck_moulded_form_design_parameter (see 4.2.30), a Bottom_moulded_form_design_parameter (see 4.2.31), a Bulb_moulded_form_design_parameter (see 4.2.32), a Propeller_moulded_form_design_parameter (see 4.2.33), a Rudder_moulded_form_design_parameter (see 4.2.35), a Thruster_moulded_form_design_parameter (see 4.2.36), or an Appendage_moulded_form_design_parameter (see 4.2.37).

NOTE - If the moulded form characteristics data differ from the information available from any geometric representation then the geometric data have precedence. The displacement and wetted outer surface are based on the moulded form. From a naval architectural viewpoint the same parameter based on the extreme form, which includes the thickness, might be more practical.

The data associated with a Moulded_form_characteristics_definition are the following:

— defined_for;

— description;

— moulded_form_displacement;

— moulded_form_outer_surface.

4.2.27.1 defined_for

The defined_for identifies the Moulded_form for which the Moulded_form_characteristics_definition is defined for. See 4.3.66 for the application assertion.

4.2.27.2 description

The description specifies the textual description of special features of the moulded form.

4.2.27.3 moulded_form_displacement

The moulded_form_displacement measures the wetted displacement of the individual Moulded_form.

4.2.27.4 moulded_form_outer_surface

The moulded_form_outer_surface measures the wetted outer surface of the individual Moulded_form.

4.2.28 Hull_moulded_form_design_parameter

Hull_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that contain dimensions and ratios of the ship hull (see Figure 21).

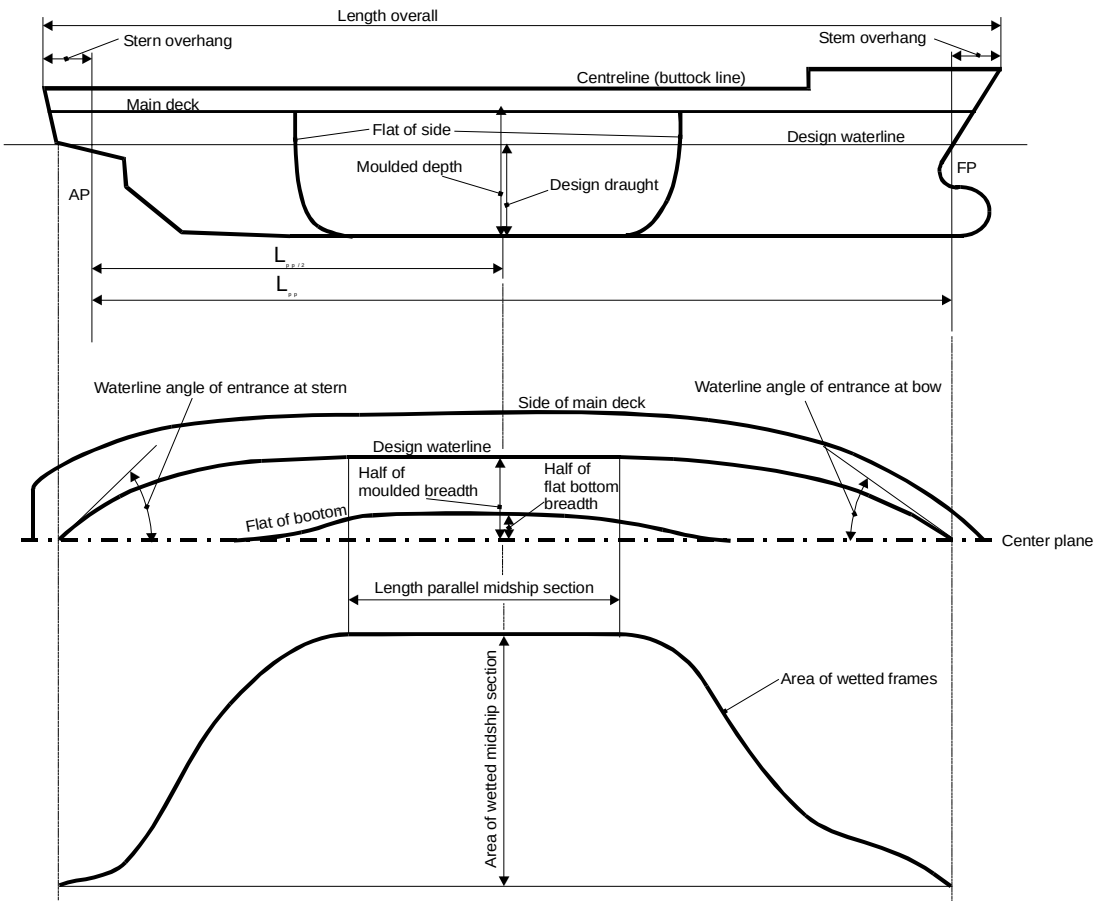


Figure 21 - Ship hull dimensions

The data associated with a Hull_moulded_form_design_parameter are the following:

— aft_end_of_flat_of_side;

— aft_end_of_parallel_midbody_at_design_draught;

— block_coefficient;

— front_end_of_flat_of_side;

— front_end_of_parallel_midbody_at_design_draught;

— gunwale_radius;

— hull_breadth;

— hull_depth;

— hull_design_draught;

— hull_length_pp;

— length_of_flat_of_side;

— length_of_parallel_midbody_at_design_draught;

— max_frame_section_area_location;

— max_wetted_frame_section_area;

— midship_tumble_data;

— prismatic_coefficient;

— waterline_angle_of_entrance_at_bow;

— waterline_angle_of_entrance_at_stern.

4.2.28.1 aft_end_of_flat_of_side

The aft_end_of_flat_of_side specifies the longitudinal distance from the Aft Perpendicular to the aft end of the flat of side, where the ship hull is of vertical shape.

4.2.28.2 aft_end_of_parallel_midbody_at_design_draught

The aft_end_of_parallel_midbody_at_design_draught specifies the longitudinal distance from the Aft Perpendicular to the aft end of the wetted midship body with sections of constant shape.

4.2.28.3 block_coefficient

The block_coefficient specifies the ratio of the moulded displacement volume to the volume of a block that has its length equal to the length_between_perpendiculars, its breadth equal to the maximum immersed moulded_breadth and its depth equal to the design_draught.

4.2.28.4 front_end_of_flat_of_side

The front_end_of_flat_of_side specifies the longitudinal distance from the Aft Perpendicular to the front end of the flat of side, where the ship hull is of vertical shape.

4.2.28.5 front_end_of_parallel_midbody_at_design_draught

The front_end_of_parallel_midbody_at_design_draught specifies the longitudinal distance from the Aft Perpendicular to the front end of the wetted midship body with sections of constant shape.

4.2.28.6 gunwale_radius

The gunwale_radius is optional and if present, specifies the radius of the arc that joins the side of the hull with the upper deck. This is measured at the transverse section amidships.

4.2.28.7 hull_breadth

The hull_breadth is optional and if present, specifies the maximum breadth of the individual Ship_hull amidships and at the design_draught.

NOTE - The hull_breadth need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_breadth is identical to the moulded_breadth in the Principal_characteristics. This parameter is only necessary for special ships e.g., multi hull ships with different Ship_hull's and different hull_breadth's.

4.2.28.8 hull_depth

The hull_depth is optional and if present, specifies for each individual Ship_hull the vertical distance above the baseline to the uppermost deck where the deck joins the side of the ship.

NOTE - The hull_depth need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_depth is identical to the moulded_depth in the Principal_characteristics. This parameter is only necessary for special ships e.g., multi hull ships with different Ship_hulls and different hull_depths.

4.2.28.9 hull_design_draught

The hull_design_draught is optional and if present, specifies the draught to which the individual Ship_hull has been designed to operate.

NOTE - The hull_design_draught need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_design_draught is identical to the design_draught in the Principal_characteristics. This parameter is only necessary for special ships e.g., multi hull ships with different Ship_hulls and different hull_design_draughts.

4.2.28.10 hull_length_pp

The hull_length_pp is optional and if present, specifies the length measured from the after perpendicular to the forward perpendicular of the individual Ship_hull.

NOTE - The hull_length_pp need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_length_pp is identical to the length_between_perpendiculars in the Principal_characteristics. This parameter is only necessary for special ships e.g. multi hull ships with different Ship_hulls and different hull_length_pps.

4.2.28.11 length_of_flat_of_side

The length_of_flat_of_side specifies the length of the flat of side where the ship hull is of vertical shape.

4.2.28.12 length_of_parallel_midbody_at_design_draught

The length_of_parallel_midbody_at_design_draught specifies the length of the wetted midship body with sections of a constant shape.

4.2.28.13 max_frame_section_area_location

The max_frame_section_area_location is optional and if present, specifies the longitudinal distance of the frame section with the maximum immersed area at design draught from the aft perpendicular.

NOTE - For normal ships with parallel midship body the max_frame_section_area_location is amidships at Lpp/ 2.

4.2.28.14 max_wetted_frame_section_area

The max_wetted_frame_section_area specifies the maximum wetted area of a frame section at design draught.

NOTE - For normal shaped ship hulls this is the frame section amidships. For special shapes e.g. sailing yachts the max_wetted_frame_section_area can be a section located at 1/3 of the ships length from the stern.

4.2.28.15 midship_tumble_data

The midship_tumble_data is optional and if present, specifies specific design parameter used for the definition of the midship section. See 4.3.41 for the application assertion.

4.2.28.16 prismatic_coefficient

The prismatic_coefficient specifies the ratio of the moulded displacement volume to the volume of a prism having a length equal to the length between perpendiculars and a cross-sectional area equal to that of the cross-sectional area of the hull amidships.

4.2.28.17 waterline_angle_of_entrance_at_bow

The waterline_angle_of_entrance_at_bow is optional and if present, specifies the angle between the tangent of the horizontal projection of the hull form at the ship's bow on the design waterline and the x-axis.

4.2.28.18 waterline_angle_of_entrance_at_stern

The waterline_angle_of_entrance_at_stern is optional and if present, specifies the angle between the tangent of the horizontal projection of the hull form at the ship's stern on the design waterline and the x-axis.

4.2.29 Midship_tumble

A Midship_tumble is a specific design parameter used for the definition of the midship section (see Figure 22).

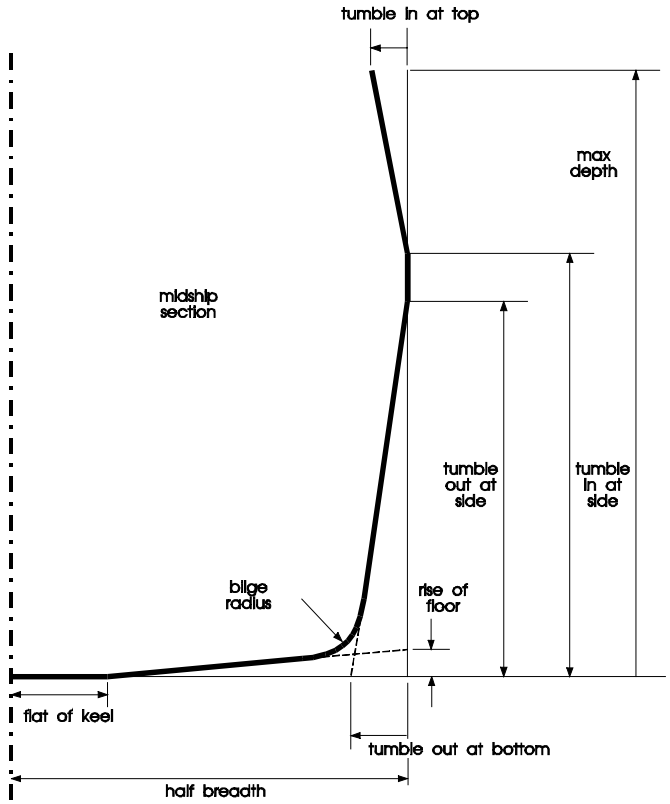


Figure 22 – Midship tumble

The data associated with a Midship_tumble are the following:

— tumble_in_at_side;

— tumble_in_at_top;

— tumble_out_at_bottom;

— tumble_out_at_side.

4.2.29.1 tumble_in_at_side

The tumble_in_at_side specifies the height of the upper extent of the midship half width above the projection of the flat of keel.

4.2.29.2 tumble_in_at_top

The tumble_in_at_top specifies the distance from the top of the midship section at the maximum depth above the base to the maximum half beam.

4.2.29.3 tumble_out_at_bottom

The tumble_out_at_bottom specifies the distance outboard on the flat of keel extended from the projection of the lower side of the midship section to the perpendicular through the maximum half beam.

4.2.29.4 tumble_out_at_side

The tumble_out_at_side specifies the height of the lower extent of the midship half width above the projection of the flat of keel.

4.2.30 Deck_moulded_form_design_parameter

Deck_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that specifies the main parameter of the deck.

The data associated with a Deck_moulded_form_design_parameter are the following:

— camber;

— defined_for;

— sheer_at_AP;

— sheer_at_FP.

4.2.30.1 camber

The camber specifies the transverse curvature of a deck at the midship section. It is measured as the difference between the deck height at the centre plane of the hull and the height of the deck at the edge where it joins the flat side of the hull.

4.2.30.2 defined_for

The defined_for identifies the Moulded_form for which the Deck_moulded_form_design_parameter are defined for. See 4.3.19 for the application assertion.

4.2.30.3 sheer_at_AP

The sheer_at_AP specifies the longitudinal curvature of the deck in the aft half of the ship. It is measured in the centre plane of the hull as the difference between the deck height amidships and the height of the deck at the aft perpendicular.

4.2.30.4 sheer_at_FP

The sheer_at_FP specifies the longitudinal curvature of the deck in the front half of the ship. It is measured in the centre plane of the hull as the difference between the deck height amidships and the height of the deck at the forward perpendicular.

4.2.31 Bottom_moulded_form_design_parameter

A Bottom_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that contains dimensions and ratios of the keel and the bottom of a Ship_hull (see Figure 23).

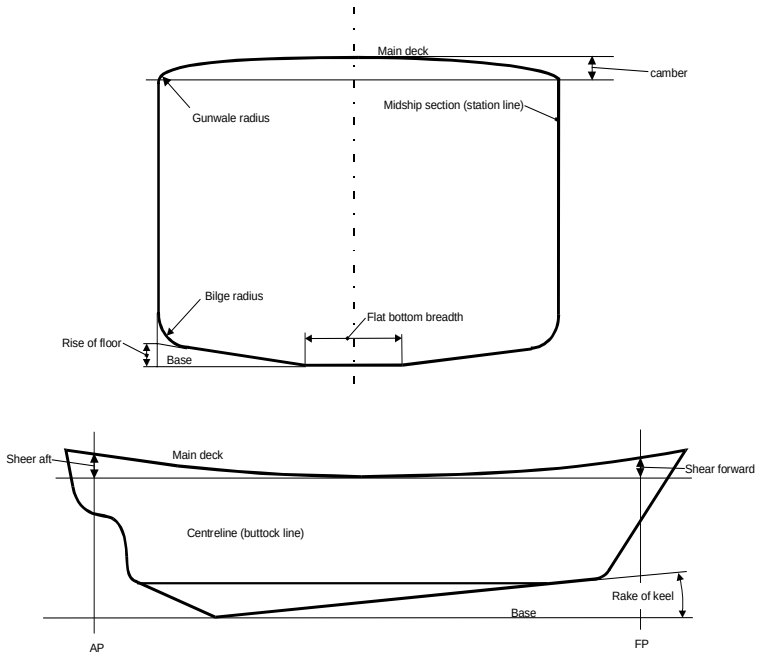


Figure 23 - Keel dimensions

The data associated with a Bottom_moulded_form_design_parameter are the following:

— aft_end_of_flat_of_bottom;

— bilge_radius;

— flat_of_bottom_breadth;

— front_end_of_flat_of_bottom;

— length_of_flat_of_bottom;

— rake_of_keel;

— rise_of_floor.

4.2.31.1 aft_end_of_flat_of_bottom

The aft_end_of_flat_of_bottom specifies the longitudinal distance from the Aft Perpendicular to the aft end of the flat bottom.

4.2.31.2 bilge_radius

The bilge_radius specifies the radius of the arc that joins the flat side of the hull to the bottom. This is measured at the transverse section amidships.

4.2.31.3 flat_of_bottom_breadth

The flat_of_bottom_breadth specifies the breadth of the flat bottom measured in transverse direction amidships.

4.2.31.4 front_end_of_flat_of_bottom

The front_end_of_flat_of_bottom specifies the longitudinal distance from the Aft Perpendicular to the front end of the flat bottom.

4.2.31.5 length_of_flat_of_bottom

The length_of_flat_of_bottom specifies the length of the flat bottom measured horizontally at the centreline.

4.2.31.6 rake_of_keel

The rake_of_keel specifies the angle between the baseline and the keel of the ship.

4.2.31.7 rise_of_floor

The rise_of_floor specifies the vertical distance from the baseline to the point where the line of the continuation of the sloped bottom meets the continuation of the line of the side of the ship. The rise_of_floor is measured at the transverse section amidships.

4.2.32 Bulb_moulded_form_design_parameter

A Bulb_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that contains dimensions and ratios of the bulb of a Ship_hull (see Figure 24).

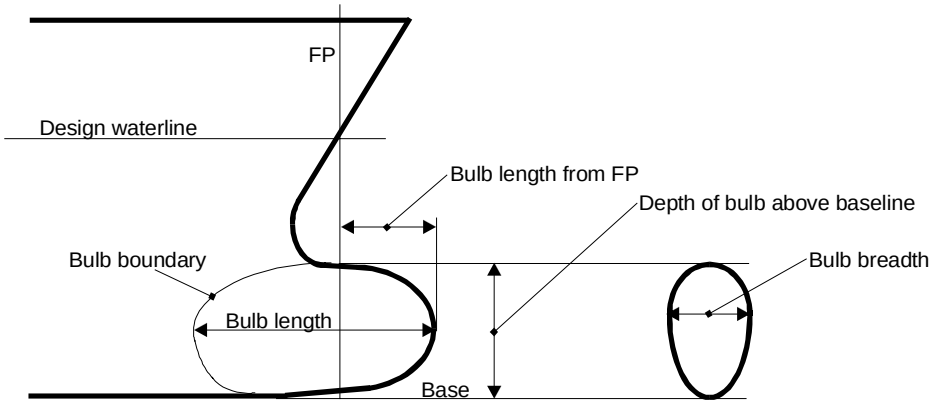


Figure 24 - Dimensions of the bulbous bow

The data associated with a Bulb_moulded_form_design_parameter are the following:

— bulb_breadth;

— bulb_breadth_pp;

— bulb_depth;

— bulb_depth_pp;

— bulb_frame_section_area_at_pp;

— bulb_length;

— bulb_length_from_pp;

— bulb_location.

4.2.32.1 bulb_breadth

The bulb_breadth specifies the maximum breadth of the bulb.

4.2.32.2 bulb_breadth_pp

The bulb_breadth_pp specifies the breadth of the bulb at the aft or forward perpendicular, depending on the location of the bulb.

4.2.32.3 bulb_depth

The bulb_depth specifies the maximum vertical distance from the baseline to the uppermost point of the bulb.

4.2.32.4 bulb_depth_pp

The bulb_depth_pp specifies the vertical distance from the baseline to the uppermost point of the bulb at the aft or forward perpendicular, depending on the location of the bulb.

4.2.32.5 bulb_frame_section_area_at_pp

The bulb_frame_section_area_at_pp specifies the surface of the bulb in the frame section at the aft or forward perpendicular, depending on the location of the bulb.

4.2.32.6 bulb_length

The bulb_length specifies the maximum length of the bulb measured horizontally at the centreline.

4.2.32.7 bulb_length_from_pp

The bulb_length_from_pp specifies the length of the bulb measured horizontally from the aft or forward perpendicular, depending on the location of the bulb.

4.2.32.8 bulb_location

The bulb_location specifies whether the bulb is located at the bow or at the stern.

The bulb_location shall be one of the following:

— bow,

— stern.

4.2.32.8.1 bow

bow: the location is at the bow of the Ship_hull.

4.2.32.8.2 stern

stern: the location is at the stern of the Ship_hull.

4.2.33 Propeller_moulded_form_design_parameter

A Propeller_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that contains dimensions and ratios of the propeller (see Figure 25).

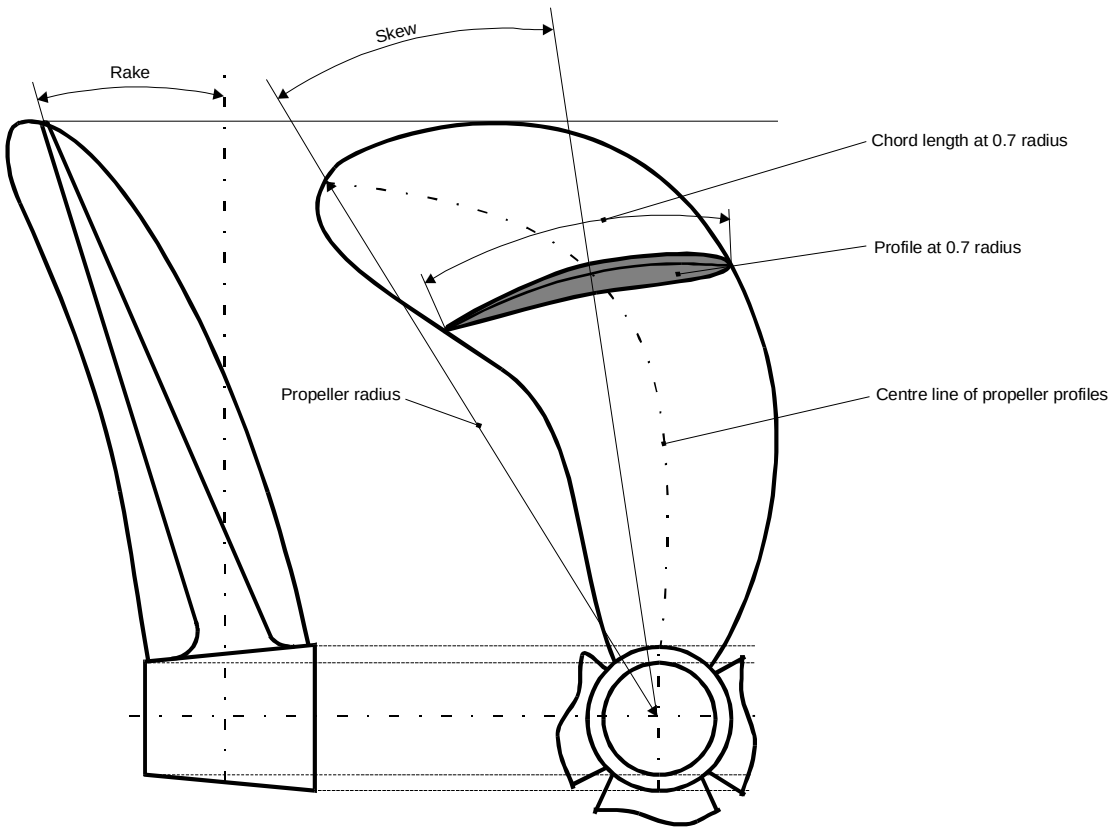


Figure 25 - Propeller dimensions

The data associated with a Propeller_moulded_form_design_parameter are the following:

— blade_mean_height;

— chord_length_at_0_7_radius;

— design_sense_of_rotation;

— expanded_area_ratio;

— hub_diameter_ratio;

— location_of_the_propeller_at_the_ship_hull;

— nominal_design_pitch_ratio;

— number_of_propeller_blades;

— propeller_diameter;

— rake;

— skew;

— thickness_at_0_7_radius;

— type_of_propeller_blades;

— type_of_propulsion.

4.2.33.1 blade_mean_height

The blade_mean_height is optional and, if present, specifies the mean height of blades for non screw propellers.

NOTE - The mean height of the blade is the largest dimension of the blade and larger than the mean breadth.

4.2.33.2 chord_length_at_0_7_radius

The chord_length_at_0_7_radius specifies the chord length of the propeller blade section located in 0.7 of the propeller radius or the mean breadth of blades for non screw propellers.

4.2.33.3 design_sense_of_rotation

The design_sense_of_rotation specifies whether the design sense of rotation for the propeller is right-handed or left-handed.

The design_sense_of_rotation shall be one of the following:

— left,

— Right.

4.2.33.3.1 left

left: the propeller rotates anti-clockwise.

4.2.33.3.2 right

right: the propeller rotates clockwise.

4.2.33.4 expanded_area_ratio

The expanded_area_ratio specifies a dimensionless parameter, which is the result of the division of the developed area of the propeller by the area of the circle described by the propeller diameter. The developed propeller area is the total area of the blades assuming no pitch, i.e. as if the blades were flattened into one plane.

4.2.33.5 hub_diameter_ratio

The hub_diameter_ratio specifies a dimensionless parameter, which is the result of the division of the diameter of the propeller hub by the propeller diameter.

4.2.33.6 location_of_the_propeller_at_the_ship_hull

The location_of_the_propeller_at_the_ship_hull is optional and if present, specifies the position of the propeller relative to the Ship_hull. See 4.3.82 for the application assertion.

4.2.33.7 nominal_design_pitch_ratio

The nominal_design_pitch_ratio specifies a dimensionless parameter, which is the result of the division of the mean design pitch by the propeller diameter.

4.2.33.8 number_of_propeller_blades

The number_of_propeller_blades specifies the number of blades on the propeller.

4.2.33.9 propeller_diameter

The propeller_diameter specifies the diameter of the largest circle that is prescribed by the rotation of the propeller blades.

4.2.33.10 rake

The rake specifies the angle between the propeller plane and the generator line.

4.2.33.11 skew

The skew specifies the angle between a straight line through the centre of the propeller and the centre of the blade profile at 0.7 propeller radius and a second straight line through the centre of the propeller and the centre of the blade profile at 1 propeller radius.

4.2.33.12 thickness_at_0_7_radius

The thickness_at_0_7_radius specifies the profile thickness of the propeller blade section located in 0.7 of the propeller radius or the maximum thickness of blades for non screw propellers.

4.2.33.13 type_of_propeller_blades

The type_of_propeller_blades specifies whether the type of the propeller blade is fixed, adjustable or controllable.

The type_of_propeller_blades shall be one of the following:

— adjustable_blade,

— controllable_blade,

— fixed_blade.

4.2.33.13.1 adjustable_blade

adjustable_blade: the propeller blades can be adjusted manually in a dry dock to meet varying operating conditions.

4.2.33.13.2 controllable_blade

controllable_blade: the propeller blades can be controlled hydraulically to vary the pitch and it is thus possible to vary thrust to the point of providing astern thrust without reversing the direction of rotation (see Figure 26).

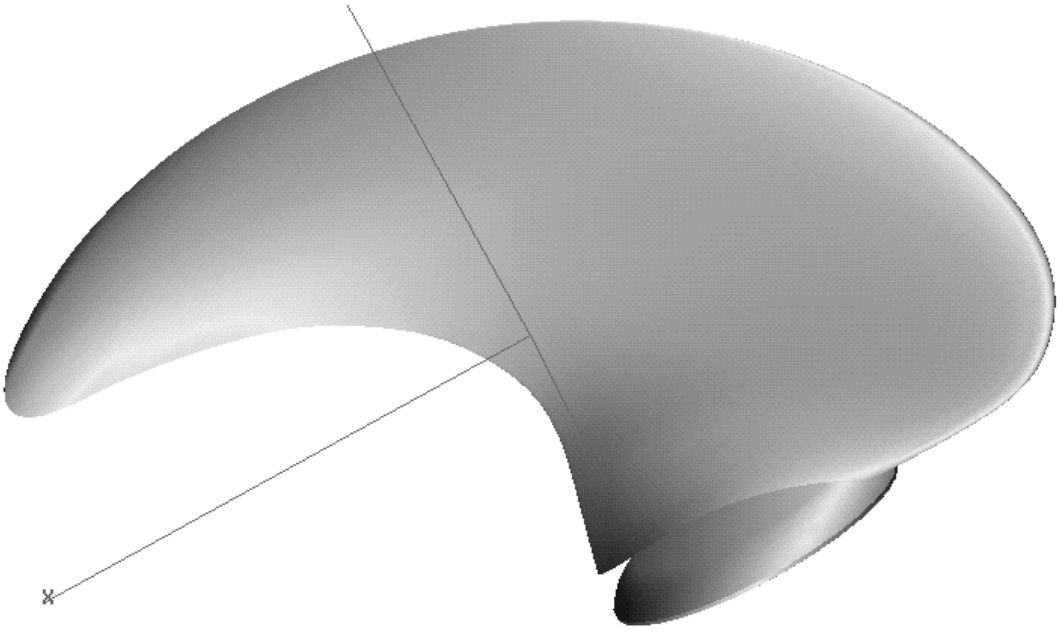


Figure 26 – Controllable propeller blade

4.2.33.13.3 fixed_blade

fixed_blade: the propeller blades have a fixed pitch (see Figure 27).

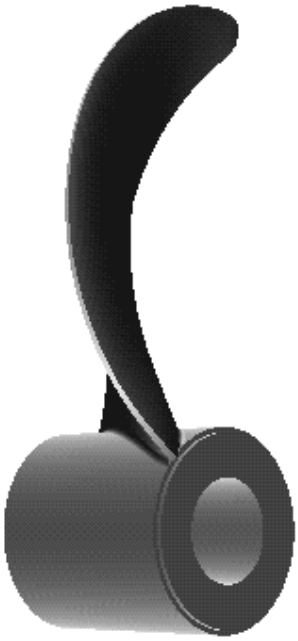


Figure 27 – Fixed propeller blade

4.2.33.14 type_of_propulsion

The type_of_propulsion specifies the individual type of propulsion system.

The type_of_propulsion shall be one of the following:

— ducted_propeller,

— paddle_wheel,

— screw_propeller,

— thruster,

— user_defined,

— vertical_axis_propeller,

— water_jet.

4.2.33.14.1 ducted_propeller

ducted_propeller: the propeller works in a duct to increase efficiency (see Figure 28).

Figure 28 – Ducted propeller

4.2.33.14.2 paddle_wheel

paddle_wheel: a more obsolete propulsion system where paddle blades rotate around a horizontal axis which is parallel to the y axis of the ship.

NOTE - This old fashioned type of propulsion is not used in modern ships.

4.2.33.14.3 screw_propeller

screw_propeller: the most common propulsion type without duct or tunnel (see Figure 29).

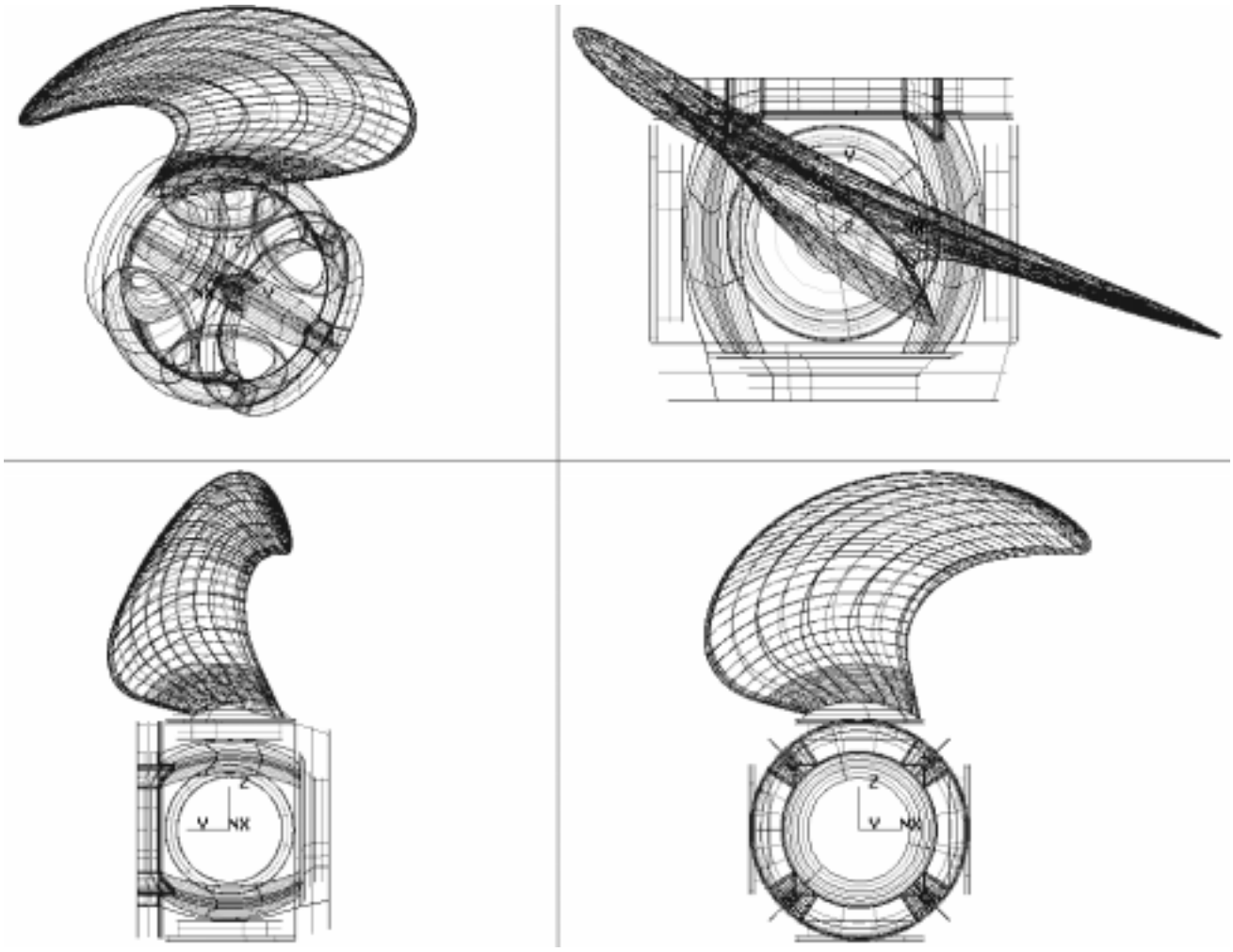


Figure 29 – Controllable screw propeller

4.2.33.14.4 thruster

thruster: the propeller works in a tunnel inside the ship hull for manoeuvring purposes (see Figure 30).

Figure 30 - Thruster

4.2.33.14.5 user_defined

user_defined: the propulsion type is user defined.

4.2.33.14.6 vertical_axis_propeller

vertical_axis_propeller: propeller wings rotating around a vertical axis parallel to the z axis of the ship. A special mechanism changes the angle of attack for each wing during rotation (see Figure 31 and Figure 32).

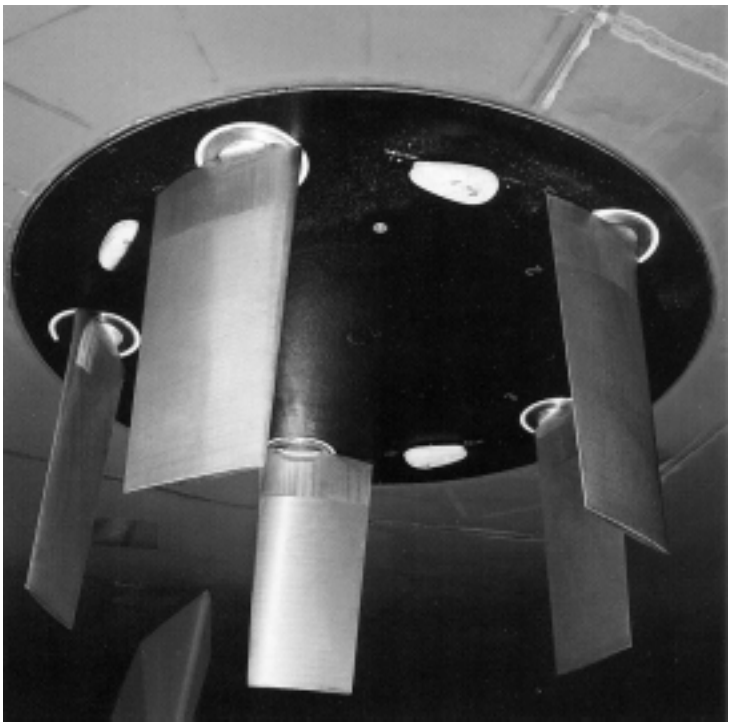


Figure 31 - Vertical axis propeller

Figure 32 - Vertical axis propeller technology

4.2.33.14.7 water_jet

water_jet: the propeller works in a tunnel inside the ship hull.

4.2.34 Propeller_location

A Propeller_location describes the position of the propeller relative to the Ship_hull (see Figure 33).

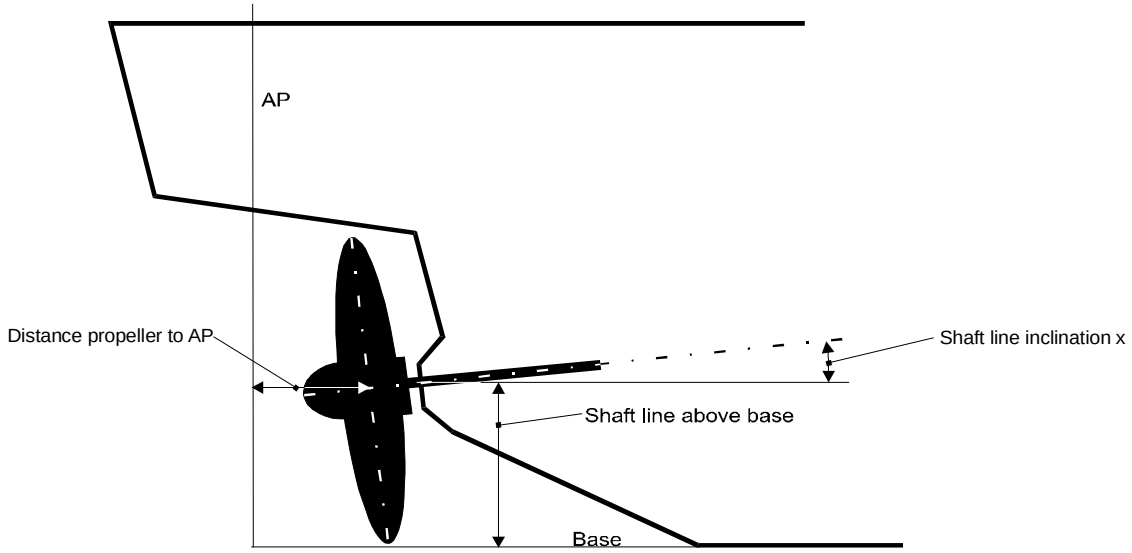


Figure 33 - Propeller location

The data associated with a Propeller_location are the following:

— propeller_location;

— shaft_line_inclination_x;

— shaft_line_inclination_y;

— shaft_line_location.

4.2.34.1 propeller_location

The propeller_location specifies the location of the centreline of the propeller shaft at the point where the propeller shaft is attached to the propeller. The x co-ordinate is measured from the aft perpendicular, the y co-ordinate is measured from the centreline and the z co-ordinate is measured from the baseline. See 4.3.80 for the application assertion.

4.2.34.2 shaft_line_inclination_x

The shaft_line_inclination_x specifies the angle between the propeller shaft and the x-axis of the Ship_hull.

4.2.34.3 shaft_line_inclination_y

The shaft_line_inclination_y specifies the angle between the propeller shaft and the y-axis of the Ship_hull.

4.2.34.4 shaft_line_location

The shaft_line_location specifies the location of the centreline of the propeller shaft at the point where the propeller shaft comes out of the Ship_hull. The x co-ordinate is measured from the aft perpendicular, the y co-ordinate is measured from the centreline and the z co-ordinate is measured from the baseline. See 4.3.81 for the application assertion.

4.2.35 Rudder_moulded_form_design_parameter

Rudder_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that contains dimensions and ratios of the rudder (see Figure 34).

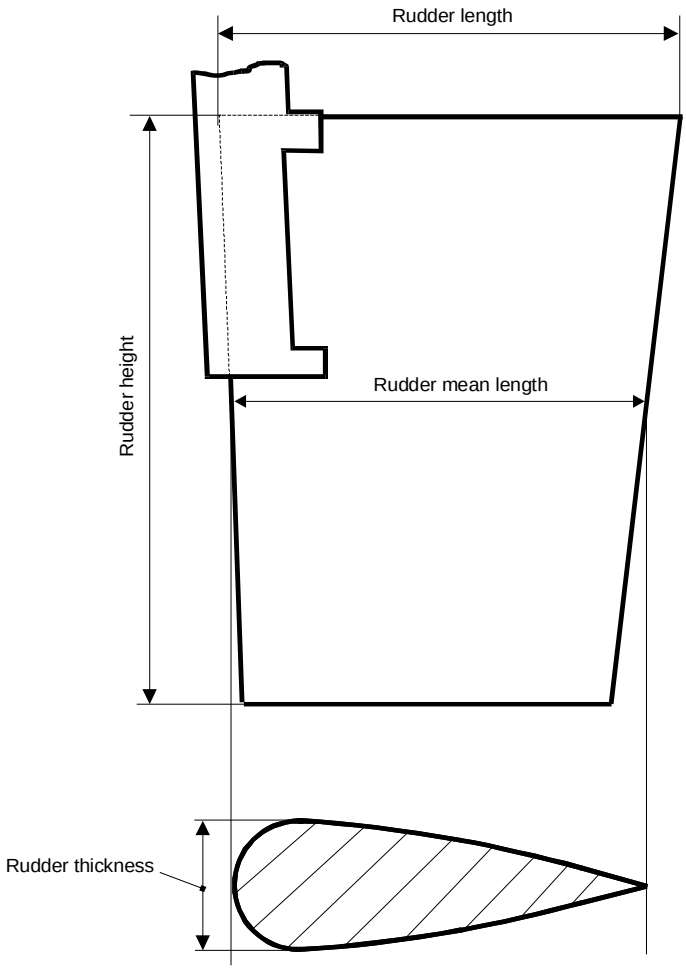


Figure 34 - Rudder dimensions

The data associated with a Rudder_moulded_form_design_parameter are the following:

— aspect_ratio;

— projected_rudder_area;

— rudder_height;

— rudder_length;

— rudder_location;

— rudder_mean_height;

— rudder_mean_length;

— rudder_thickness;

— type_of_the_rudder.

4.2.35.1 aspect_ratio

The aspect_ratio specifies the aspect ratio between the mean values of its height and length.

4.2.35.2 projected_rudder_area

The projected_rudder_area specifies the surface of the rudder in its operating position projected into the xy-plane of the ship.

4.2.35.3 rudder_height

The rudder_height specifies the height of the rudder measured from its lowest to its highest point in the operating position.

4.2.35.4 rudder_length

The rudder_length specifies the maximum length of the rudder in its operating position measured lengthways of the ship.

4.2.35.5 rudder_location

The rudder_location specifies the location of the centreline of the rudder shaft at the point where it is attached to the top of the rudder. The x co-ordinate is measured from the aft perpendicular, the y co-ordinate is measured from the centreline and the z co-ordinate is measured from the baseline. See 4.3.89 for the application assertion.

4.2.35.6 rudder_mean_height

The rudder_mean_height specifies the mean height of the rudder in its operating position.

4.2.35.7 rudder_mean_length

The rudder_mean_length specifies the mean length of the rudder in its operating position measured in the length direction of the ship.

4.2.35.8 rudder_thickness

The rudder_thickness specifies the maximum thickness of the rudder.

4.2.35.9 type_of_the_rudder

The type_of_the_rudder specifies whether it is a balanced or unbalanced rudder.

The type_of_the_rudder shall be one of the following:

— balanced,

— unbalanced.

4.2.35.9.1 balanced

balanced: the rudder area is equally distributed in front and behind the rudder turning axis in order to reduce the operating torque at the rudder stock.

4.2.35.9.2 unbalanced

unbalanced: the rudder area is located behind the rudder axis of turning.

4.2.36 Thruster_moulded_form_design_parameter

A Thruster_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.27) that contains the dimensions and ratios of the thruster tunnel and propeller.

The data associated with a Thruster_moulded_form_design_parameter are the following:

— geometric_thruster_location;

— thruster_location;

— thruster_propeller_parameter;

— thruster_tunnel_diameter;

— thruster_tunnel_max_length;

— thruster_tunnel_min_length.

4.2.36.1 geometric_thruster_location

The geometric_thruster_location specifies the location of the centreline of the thruster tunnel. The x co-ordinate is measured from the aft perpendicular, the y co-ordinate is measured from the centreline and the z co-ordinate is measured from the baseline. See 4.3.117 for the application assertion.

4.2.36.2 thruster_location

The thruster_location specifies whether the thruster is located at the bow or at the stern.

The thruster_location shall be one of the following:

— bow,

— stern.

4.2.36.2.1 bow

bow: the location is at the bow of the Ship_hull.

4.2.36.2.2 stern

stern: the location is at the stern of the Ship_hull.

4.2.36.3 thruster_propeller_parameter

The thruster_propeller_parameter specifies the dimensions and ratios of the propeller in the thruster tunnel (see Figure 35). See 4.3.118 for the application assertion.

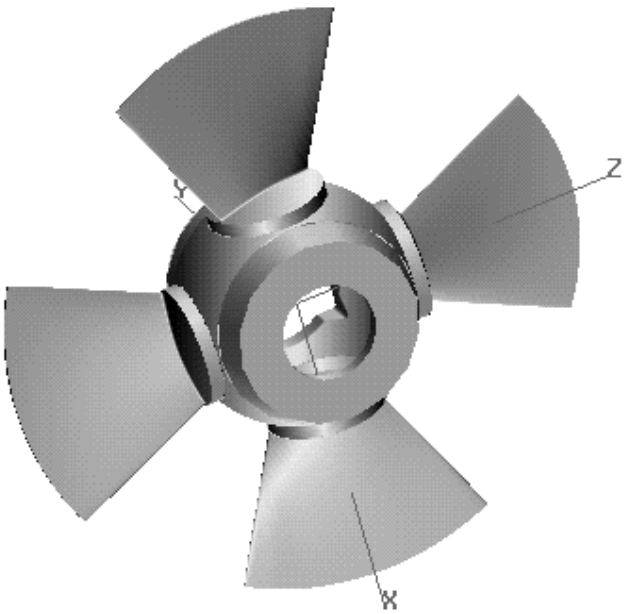


Figure 35 – Thruster propeller

4.2.36.4 thruster_tunnel_diameter

The thruster_tunnel_diameter specifies the diameter of the thruster tunnel.

4.2.36.5 thruster_tunnel_max_length

The thruster_tunnel_max_length specifies the maximum length of the thruster tunnel when it intersects with the Ship_hull.

4.2.36.6 thruster_tunnel_min_length

The thruster_tunnel_min_length specifies the minimum length of the thruster tunnel when it intersects with the Ship_hull.

4.2.37 Appendage_moulded_form_design_parameter

An Appendage_moulded_form_design_parameter is a type of Moulded_form_characteristics_ definition (see 4.2.27) that contains the dimensions and ratios of any ship hull appendage.

The data associated with an Appendage_moulded_form_design_parameter are the following:

— appendage_breadth;

— appendage_depth;

— appendage_length;

— type_of_appendage.

4.2.37.1 appendage_breadth

The appendage_breadth specifies the breadth of the appendage.

NOTE - The breadth is the second largest dimension of the appendage.

4.2.37.2 appendage_depth

The appendage_depth specifies the depth of the appendage.

NOTE - The depth is the smallest dimension of the appendage.

4.2.37.3 appendage_length

The appendage_length specifies the length of the appendage.

NOTE - The length is the maximum dimension of the appendage.

4.2.37.4 type_of_appendage

The type_of_appendage identifies the type of the appendage.

The type_of_appendage shall be one of the following:

— bilge_keel,

— passive_fin,

— shaft_bossings,

— shaft_strut,

— skeg,

— spray_rail.

4.2.37.4.1 bilge_keel

bilge_keel: external fin at the round of the bilge to reduce rolling (see Figure 36).

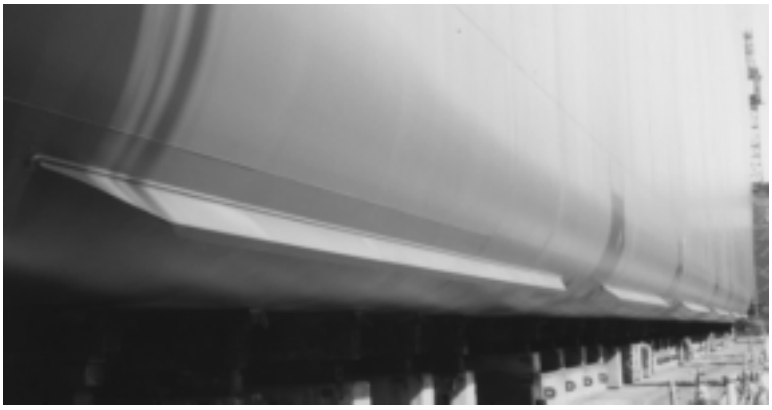


Figure 36 – Bilge keel

4.2.37.4.2 passive_fin

passive_fin: a fixed fin attached the ship hull to change the flow in this area.

4.2.37.4.3 shaft_bossings

shaft_bossings: curved portion of the ship hull that surrounds and supports the propeller shaft (see Figure 37).

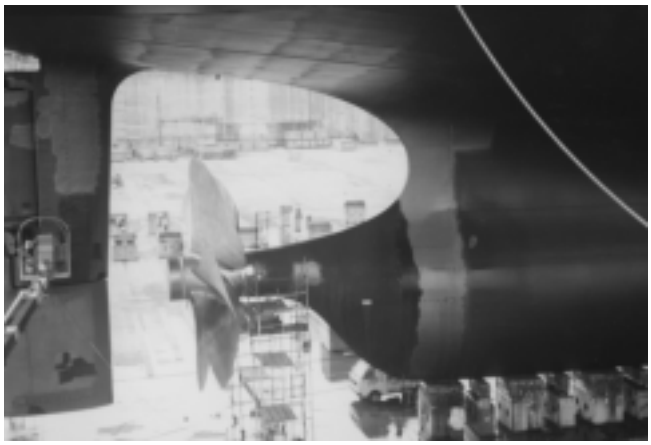


Figure 37 - Shaft bossings

4.2.37.4.4 shaft_strut

shaft_strut: a bearing stool for the part of the propeller shaft outside the hull (normally for twin or multi-screw-arrangements) (see Figure 38 and Figure 39).

Figure 38 – Shaft strut

Figure 39 – Shaft struts

4.2.37.4.5 skeg

skeg: finlike projection on the bottom of the ship hull to support the lower edge of the rudder.

4.2.37.4.6 spray_rail

spray_rail: is a ledge on the ship hull to keep away any spray water.

IV UoF hydrostatics

4.2.38 Hydrostatic_definition

A Hydrostatic_definition is a type of Design_definition (see 4.2.90) that specifies a design_definition, which describes a relationship between a Ship_moulded_form, a set of modifications to the displacement of that Ship_moulded_form and different Hydrostatic_tables. The modifications will be due to different combinations of inlets and appendages with the Ship_moulded_form.

The data associated with a Hydrostatic_definition are the following:

— defined_for;

— displacement_change_description;

— displacement_changes;

— representations.

4.2.38.1 defined_for

The defined_for specifies the Ship for which the Hydrostatic_definition is defined. See 4.3.44 for the application assertion.

4.2.38.2 displacement_change_description

The displacement_change_description is optional and if present, specifies the textual description of the change in the displacement.

4.2.38.3 displacement_changes

The displacement_changes specify the list of Displacement_operations that detail additions or subtractions to the Ship_moulded_form due to combinations of inlets and appendages. There may be more than one displacement_operations for a hydrostatic_definition. See 4.3.42 for the application assertion.

EXAMPLE - For a hull with a bow thruster tunnel, the hydrostatic properties will be modified by the inclusion of the thruster tunnel into the ship hull. Such an inclusion would be a displacement change.

4.2.38.4 representations

The representations specify the Hydrostatic_tables, which represent the Hydrostatic_definition. See 4.3.43 for the application assertion.

4.2.39 Displacement_operation

A Displacement operation is either an addition_of_moulded_form or a Subtraction_of_moulded_form. A Displacement operation is a calculation for modifying the displacement of a Ship_moulded_form.

The modification is described by the, addition or subtraction of a Moulded_form. A Displacement_operation may be an Addition_of_moulded_form (see 4.2.40) or a Subtraction_of_moulded_form (see 4.2.41).

The data associated with a Displacement_operation is the following:

— displacement_of_moulded_form_to_add_or_subtract.

The displacement_of_moulded_form_to_add_or_subtract specifies the Moulded_form that is either added or subtracted to the Ship_moulded_form. The displacement of the Moulded_form can be found in the specific moulded form characteristics. See 4.3.32 for the application assertion.

4.2.40 Addition_of_moulded_form

An Addition_of_moulded_form is atype of Displacement_operation (see 4.2.39) that specifies a type of displacement_operation whereby the displacement of the Ship_moulded_form will be increased by the displacement of the new Moulded_form.

4.2.41 Subtraction_of_moulded_form

A Subtraction_of_moulded_form is a type of Displacement_operation (see 4.2.39) that specifies a type of displacement_operation whereby the displacement of the Ship_moulded_form will be reduced by the displacement of the new Moulded_form.

4.2.42 Hydrostatic_table

A Hydrostatic_table is a type of Representation that defines the hydrostatic properties and values for a given Hydrostatic_definition. A Hydrostatic_table consists of a list of Hydrostatic_properties_for_constant_floating_position, where each entry covers all hydrostatic properties of a ship for a different Floating_position. Because the hydrostatic properties are related to the extreme form, a uniform shell thickness is used to generate the extreme form from the moulded form.

NOTE - Representation is a STEP resource entity defined in ISO 10303-43.

The data associated with a Hydrostatic_table are the following:

— items;

— mean_shell_thickness;

— properties_in_the_hydrostatic_table.

4.2.42.1 items

The items specify the Hydrostatic_properties_for_constant_floating_position that describes all hydrostatic properties for a Ship, for a specific Floating_position. See 4.3.50 for the application assertion.

4.2.42.2 mean_shell_thickness

The mean_shell_thickness specifies the real value for the average thickness of the shell plating, which may be used to define the related extreme form for the hydrostatic properties for the ship including the thickness (see Figure 40).

Figure 40 - Plate thickness

4.2.42.3 properties_in_the_hydrostatic_table

The properties_in_the_hydrostatic_table specify the list of Hydrostatic_properties for the Hydrostatic_table. See 4.3.51 for the application assertion.

4.2.43 Hydrostatic_property

A Hydrostatic_property is a hydrostatic parameter used to describe the hydrostatic properties of a ship. All hydrostatic properties are dependent on the Floating_position of the ship.

The data associated with a Hydrostatic_property are the following:

— property_type;

— property_unit.

4.2.43.1 property_type

The property_type specifies the type of parameter of interest.

The property_type shall be one of the following:

— centre_of_buoyancy,

— centre_of_flotation,

— centre_of_lateral_resistance,

— longitudinal_metacentric_height,

— longitudinal_second_moment_of_area_of_waterplane,

— mid_ship_section_area,

— moment_for_unit_change_of_trim,

— transverse_metacentric_height,

— transverse_second_moment_of_area_of_waterplane,

— waterplane_area,

— wetted_surface_area.

4.2.43.1.1 centre_of_buoyancy

centre_of_buoyancy: the centroid of the volume displaced by the ship (called B), measured in the global co-ordinate system.

4.2.43.1.2 centre_of_flotation

centre_of_flotation: the geometric centroid of the waterplane area of the hull moulded form, measured in the global co-ordinate system.

4.2.43.1.3 centre_of_lateral_resistance

centre_of_lateral_resistance: the location of the centre of all forces caused by the lateral resistance of the ship if it moves sideways. This location is a property of the shape of the hull moulded form and is measured in the global co-ordinate system. This location is of interest for instance for the design of thrusters at the bow and the stern of the ship.

4.2.43.1.4 longitudinal_metacentric_height

longitudinal_metacentric_height: the vertical height from the centre of gravity (called G) of the ship to the longitudinal meta-centre. The longitudinal meta-centre is defined as the intersection point of the plane amidships with the line of action of the correcting force through the centre of buoyancy that results from a pitch angle.

4.2.43.1.5 longitudinal_second_moment_of_area_of_waterplane

longitudinal_second_moment_of_area_of_waterplane: the second moment of the area of the waterplane about the transverse axis.

4.2.43.1.6 mid_ship_section_area

mid_ship_section_area: the area of the immersed part of the cross section of a hull moulded form at amidships.

4.2.43.1.7 moment_for_unit_change_of_trim

moment_for_unit_change_of_trim: the moment about the transverse axis required to change the difference between fore and aft draughts by one unit.

4.2.43.1.8 transverse_metacentric_height

transverse_metacentric_height: the vertical height from the centre of gravity (called G) of the ship to the transverse meta-centre (see Figure 41). The transverse meta-centre is defined as the intersection

point of the centre plane with the line of action of the correcting force through the centre of buoyancy that results from a roll angle (see Figure 41).

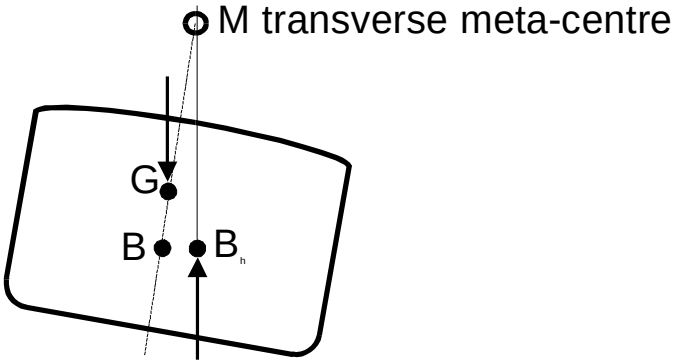


Figure 41 – Transverse meta- centre

4.2.43.1.9 transverse_second_moment_of_area_of_waterplane

transverse_second_moment_of_area_of_waterplane: the second moment of the area of the of the waterplane about the longitudinal axis.

4.2.43.1.10 waterplane_area

waterplane_area: the enclosed area of the plane formed by the intersection of the hull moulded form with the surface of the water.

4.2.43.1.11 wetted_surface_area

wetted_surface_area: the area of the outer surface of the ship that is in contact with water.

4.2.43.2 property_unit

The property_unit specifies the measurement unit of the Hydrostatic_property. See 4.3.49 for the application assertion.

4.2.44 Hydrostatic_properties_for_constant_floating_position

A Hydrostatic_properties_for_constant_floating_position is a type of Representation_item that defines the hydrostatic properties for one specific Floating_position.

NOTE - Representation_item is a STEP resource entity defined in ISO 10303-43.

The data associated with a Hydrostatic_properties_for_constant_floating_position are the following:

— definition_of_floating_position;

— hydrostatic_property_values;

— related_hydrostatic_table.

4.2.44.1 definition_of_floating_position

The definition_of_floating_position specifies the Floating_position for which the Hydrostatic_property_values are valid. See 4.3.46 for the application assertion.

4.2.44.2 hydrostatic_property_values

The hydrostatic_property_values specifies the hydrostatic properties for the ship for the specific Floating_position. See 4.3.47 for the application assertion.

4.2.44.3 related_hydrostatic_table

The related_hydrostatic_table specifies the set of Hydrostatic_tables with which the Hydrostatic_properties_for_constant_floating_position are associated. See 4.3.48 for the application assertion.

4.2.45 Floating_position

A Floating_position is the draught and attitude of the ship when immersed and the resulting displacement volume.

The data associated with a Floating_position are the following:

— angle_of_heel;

— angle_of_trim;

— breadth_of_waterline;

— draught_at_amidships;

— length_of_waterline;

— moulded_form_displacement.

4.2.45.1 angle_of_heel

The angle_of_heel specifies the angle of rotation around the x-axis of the ship measured in radians and measured on a line parallel to the global y-axis and the waterplane. The angle_of_heel is equal to zero when the centreplane is perpendicular to the waterplane. The angle_of_heel has positive values if the starboard side of the ship moves down.

4.2.45.2 angle_of_trim

The angle_of_trim specifies the angle of rotation around the y-axis of the ship measured in radians and measured on a line parallel to the global x-axis and the waterplane. The angle_of_trim is equal to zero when the transverse cross-section is perpendicular to the waterplane. The angle_of_trim has positive values if the bow of the ship moves up.

4.2.45.3 breadth_of_waterline

The breadth_of_waterline specifies the breadth of the current waterline.

4.2.45.4 draught_at_amidships

The draught_at_amidships specify the distance from the operating waterplane to the moulded bottom of the ship, measured perpendicular at the centreline on the transverse cross-section amidships.

4.2.45.5 length_of_waterline

The length_of_waterline specifies the length of the current waterline.

4.2.45.6 moulded_form_displacement

The moulded_form_displacement specifies the wetted displacement of the ship.

4.2.46 Hydrostatic_property_value

A Hydrostatic_property_value is a hydrostatic parameter, which describes a specific hydrostatic property for a ship. A Hydrostatic_property_value is dependent on the Floating_position of the ship. A Hydrostatic_property_value may be a Hydrostatic_position_value (see 4.2.47) or a Hydrostatic_scalar_value (see 4.2.48).

4.2.47 Hydrostatic_position_value

A Hydrostatic_position_value is a type of Hydrostatic_property_value (see 4.2.46) that specifies the position of a hydrostatic property value in the global coordinate system. When applicable, a Hydrostatic_property is said to act through its Hydrostatic_position_value.

The data associated with a Hydrostatic_position_value is the following:

— position.

The position specifies the position in the global co-ordinate system through which the Hydrostatic_property acts. See 4.3.45 for the application assertion.

4.2.48 Hydrostatic_scalar_value

A Hydrostatic_scalar_value is a type of Hydrostatic_property_value (see 4.2.46) that specifies a Hydrostatic_property_value that is measured by a scalar.

The data associated with a Hydrostatic_scalar_value is the following:

— scalar.

The scalar specifies the value for the Hydrostatic_property.

4.2.49 Stability_definition

A Stability_definition is a type of Design_definition (see 4.2.90) that defines the stability properties for a given intact ship. The results are defined in a tabular form for different loading conditions and represent the righting arms and the centre of buoyancy for different heel angles.

NOTE - Damage stability will be defined in a special type of Stability_definition.

The data associated with a Stability_definition are the following:

— defined_for;

— representations.

4.2.49.1 defined_for

The defined_for specifies the Ship for which the Stability_definition is defined. See 4.3.108 for the application assertion.

4.2.49.2 representations

The representations specify the Stability_table, which the Stability_definition represents. See 4.3.109 for the application assertion.

4.2.50 Stability_table

A Stability_table is a type of Representation that defines the stability properties for a given intact or damaged ship, depending on the stability definition. The results are defined in a tabular form and represent the righting arms and the centre of buoyancy for different heel angles for one starting Floating_position (see Figure 42).

NOTE - Representation is a STEP resource entity defined in ISO 10303-43.

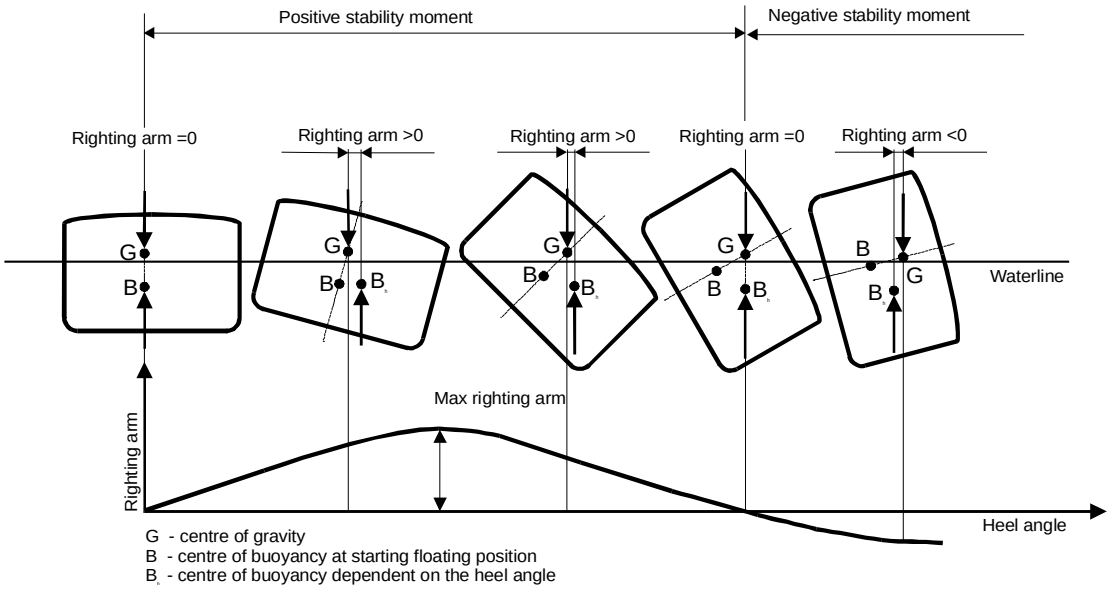


Figure 42 - Form stability

The data associated with a Stability_table are the following:

— items;

— mean_shell_thickness.

4.2.50.1 items

The items specify the stability properties, which are the righting arms and the centre of buoyancy for different heel angles for one starting Floating_position. See 4.3.115 for the application assertion.

4.2.50.2 mean_shell_thickness

The mean_shell_thickness specifies the real value for the average thickness of the shell plating, which may be used to define the related extreme form for the stability properties for the ship including the thickness.

4.2.51 Stability_properties_for_one_floating_position

A Stability_properties_for_one_floating_position is a type of Representation_item that represents the collection of values and data specific to stability calculations for a particular Floating_position for a Ship_moulded_form.

NOTE - Representation_item is a STEP resource entity defined in ISO 10303-43.

The data associated with a Stability_properties_for_one_floating_position are the following:

— centre_of_gravity_above_keel;

— definition_of_starting_floating_position;

— related_stability_table;

— stability_properties_for_different_angles_of_heel.

4.2.51.1 centre_of_gravity_above_keel

The centre_of_gravity_above_keel specifies the location for the ship's centre of gravity. It will be presumed that the location for the ship's centre of gravity will not change for different heel angles. The x co-ordinate is measured from the aft perpendicular, the y co-ordinate is measured from the centreline and the z co-ordinate is measured from the baseline. See 4.3.110 for the application assertion.

4.2.51.2 definition_of_starting_floating_position

The definition_of_starting_floating_position specifies the Floating_position of the ship for specific loading conditions, which is used as the starting point for the calculations of the stability properties. The Floating_position covers the draught, trim, heel, and corresponding volume of displacement for the ship. See 4.3.111 for the application assertion.

4.2.51.3 stability_properties_for_different_angles_of_heel

The stability_properties_for_different_angles_of_heel specifies all the stability properties, which are valid for the particular Definition_of_starting_floating_position. See 4.3.112 for the application assertion.

4.2.51.4 related_stability_table

The related_stability_table specifies the Stability_tables with which the Stability_properties_for_one_floating_position are associated. See 4.3.113 for the application assertion.

4.2.52 Stability_property

A Stability_property defines all the necessary properties for the stability calculations for a Ship_moulded_form with specific loading conditions.

The data associated with a Stability_property are the following:

— angle_of_heel;

— centre_of_buoyancy;

— righting_arm.

4.2.52.1 angle_of_heel

The angle_of_heel specifies the angle of rotation of the ship around the x-axis. The angle_of_heel has positive values if the starboard side of the ship moves down.

4.2.52.2 centre_of_buoyancy

The centre_of_buoyancy specifies the location of the volumetric centre of the submerged Ship_moulded_form volume. The x co-ordinate is measured from the aft perpendicular, the y co-ordinate is measured from the centreline and the z co-ordinate is measured from the baseline. See 4.3.114 for the application assertion.

4.2.52.3 righting_arm

The righting_arm specifies the distance between a perpendicular line through the centre_of_gravity_above_keel and a perpendicular line through the centre_of_buoyancy. Both lines are taken orthogonal to the heeled waterline.

V UoF offset_table_representations

4.2.53 Offset_table_shape_representation

An Offset_table_shape_representation is a type of Moulded_form_shape_representation (see 4.2.7) which describes the shape of the Moulded_form using offset points.

The data associated with an Offset_table_shape_representation is the following:

— items.

The items specify the set of Offset_point_table_models that build the Offset_table_shape_representation. See 4.3.75 for the application assertion.

4.2.54 Offset_point_table_model

An Offset_point_table_model is a type of Geometric_representation_item that represents the shape of the Moulded_form only by a set of Ship_points. The Ship_points are listed in sections.

NOTE 1 - Geometric_representation_item is a STEP resource entity defined in ISO 10303-42.

NOTE 2 - Normally the sections in the Offset_point_table_model are 2d sections. In special circumstances there might be a need for using 3d sections. Then the Offset_point_table_type should be user_defined_table and the sections of the offset table representing 3d curves and can be of any 3d shape.

The data associated with an Offset_point_table_model are the following:

— offset_point_table_sections;

— offset_point_table_type.

4.2.54.1 offset_point_table_sections

The offset_point_table_sections specify the list of sections that detail the separate sections of points in the offset table. See 4.3.74 for the application assertion.

EXAMPLE - A common example can be an offset table with waterline offsets. Then all points in a single section represent one waterline.

4.2.54.2 offset_point_table_type

The offset_point_table_type specifies the type of the offset table and the type of sections where the offset points are located.

The offset_point_table_type shall be one of the following:

— buttock_table,

— station_table,

— user_defined_table,

— waterline_table.

4.2.54.2.1 buttock_table

buttock_table: specifies that the offset points in the offset table are points located on buttocks in the xz plane.

4.2.54.2.2 station_table

station_table: specifies that the offset points in the offset table are points located on stations in the yz plane.

4.2.54.2.3 user_defined_table

user_defined_table: the section for the offset points is not specified.

4.2.54.2.4 waterline_table

waterline_table: specifies that the offset points in the offset table are points located on waterlines in the xy plane.

4.2.55 Section_of_offset_point_table

A Section_of_offset_point_table is an ordered list of Ship_points. This is similar to a Ship_curve representated by a polyline.

The data associated with a Section_of_offset_point_table are the following:

— for_table;

— section_identifier;

— section_points.

4.2.55.1 for_table

The for_table specifies the associated Offset_point_table_models for the Section_of_offset_point_table. See 4.3.90 for the application assertion.

4.2.55.2 section_identifier

The section_identifier specifies the context specific identification given to the section of the offset table.

4.2.55.3 section_points

The section_points specify the list of Ship_points representing the offsets of the shape this section. See 4.3.91 for the application assertion.

VI UoF wireframe_representations

4.2.56 Wireframe_shape_representation

A Wireframe_shape_representation is a type of Moulded_form_shape_representation (see4.2.7) that combines topological and geometric information together to describe a wireframe as a set of wire_shells (see Figure 43).

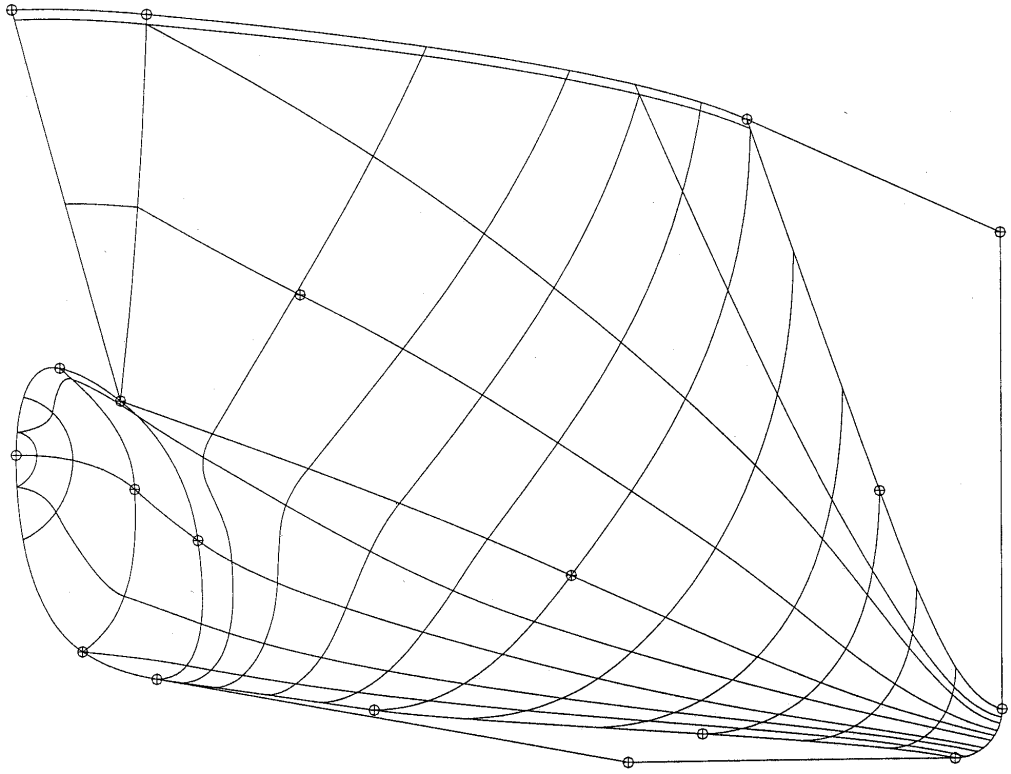


Figure 43 - Wireframe

The data associated with a Wireframe_shape_representation is the following:

— items.

The items specify the set of shell_based_surface_models that build the Wireframe_shape_representation. See 4.3.126 for the application assertion.

4.2.57 Ship_moulded_form_wireframe_model

A Ship_moulded_form_wireframe_model is a type of Shell_based_wireframe_model with additional information for a ship specific wireframe representation. The Ship_moulded_form_wireframe_model describes a wireframe, which is defined by two intersecting ship curve sets and additional bounding curves. Most of the Ship_curves are running through the entire wireframe from one end to the other end. The additional information in a Ship_moulded_form_wireframe_model is the reference to the defining Ship_curves.

NOTE - Shell_based_wireframe_model is a STEP resource entity defined in ISO 10303-42.

EXAMPLE - A typical Ship_moulded_form_wireframe_model can be formed by a list of station curves and a list of waterlines.

The data associated with a Ship_moulded_form_wireframe_model is the following:

— mesh_definition_ship_curves.

The mesh_definition_ship_curves specify the defining Ship_curves, which are intersecting at the Knots and form a ship specific wireframe. See 4.3.100 for the application assertion.

4.2.58 Ship_curve_segment

A Ship_curve_segment is a type of Edge_curve with additional information for a ship specific wireframe representation. The Ship_curve_segment is one of the bounding curves of a cell in a wireframe with information about the Ship_curve to which the Ship_curve_segment belongs.

NOTE - Edge_curve is a STEP resource entity defined in ISO 10303-42.

The data associated with a Ship_curve_segment is the following:

— part_of_ship_curve.

The part_of_ship_curve specifies the Ship_curve that the Ship_curve_segment is part of. See 4.3.94 for the apllication assertion.

4.2.59 Knot

A Knot is a type of Vertex_point with additional information for a ship specific wireframe representation. The Knot is the corner point of a cell in a wireframe with information about the intersecting ship curves in this point and information about the tangent in this point.

NOTE - Vertex_point is a STEP resource entity defined in ISO 10303-42.

The data associated with a Knot are the following:

— intersecting_ship_curves;

— tangent_information.

4.2.59.1 intersecting_ship_curves

The intersecting_ship_curves specifies the reference to the ship curves, which are intersecting in the Knot. See 4.3.62 for the application asertion.

4.2.59.2 tangent_information

The tangent_information is optional and if present, specifies the direction and the magnitude of the tangent in the Knot. This information can be useful for systems, which do not have sufficient curve geometry information. This information can be derived from the tangent information. See 4.3.63 for the application asertion.

VII UoF surface_representations

4.2.60 Surface_shape_representation

A Surface_shape_representation is a type of Moulded_form_shape_representation (see 4.2.7) that combines topological and geometric information together to describe a surface model as a set of connected surface patches (see Figure 44).

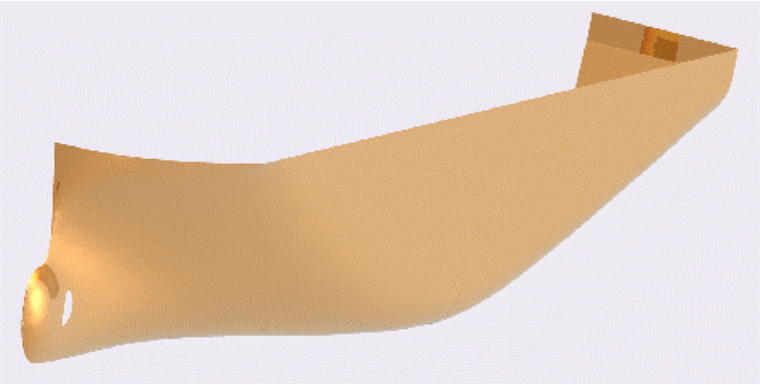


Figure 44 - Surface representation

The data associated with a Surface_shape_representation is the following:

— items.

The items specify a set of face_based_surface_models that build the Surface_shape_representation.See 4.3.116 for the application assertion.

VIII UoF basic_geometry

4.2.61 Ship_point

A Ship_point is a type of Moulded_form_representation_item (see 4.2.12) that represents a point with associated naval architectural information (see Figure 45).

EXAMPLE - A Ship_point may be a knuckle point that has a location in the global co-ordinate system.

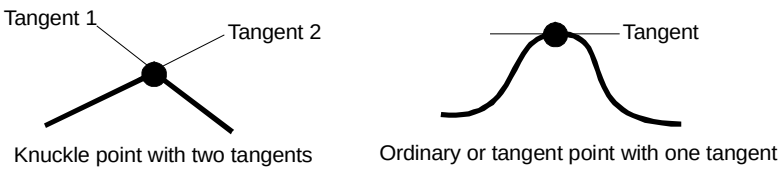


Figure 45 - Ship points

The data associated with a Ship_point are the following:

— point_class;

— point_shape.

4.2.61.1 point_class

The point_class specifies the naval architectural categories for the Ship_point based on the behaviour of curves passing through the Ship_point.

The point_class shall be one of the following:

— knuckle,

— ordinary,

— tangent,

— unspecified.

4.2.61.1.1 knuckle

knuckle: a curve passing through the point will have a discontinuous tangent either side of the point

4.2.61.1.2 ordinary

ordinary: a point of smooth tangency for one or more curves that pass through

4.2.61.1.3 tangent

tangent: a point where curves passing through have a specified tangent

4.2.61.1.4 unspecified

unspecified: a point where no tangency information is known

4.2.61.2 point_shape

The point_shape specifies the underlying geometric definition of the Ship_point See 4.3.102 for the apllication assertion.

4.2.62 Ship_curve

A Ship_curve is a type of Moulded_form_representation_item (see 4.2.12) that represents a curve with associated naval architectural information (see Figure 46). A Ship_curve may be a Ship_curve_with_spacing_position (see 4.2.63).

EXAMPLE - A waterline that has a geometry defined by a B-spline curve.

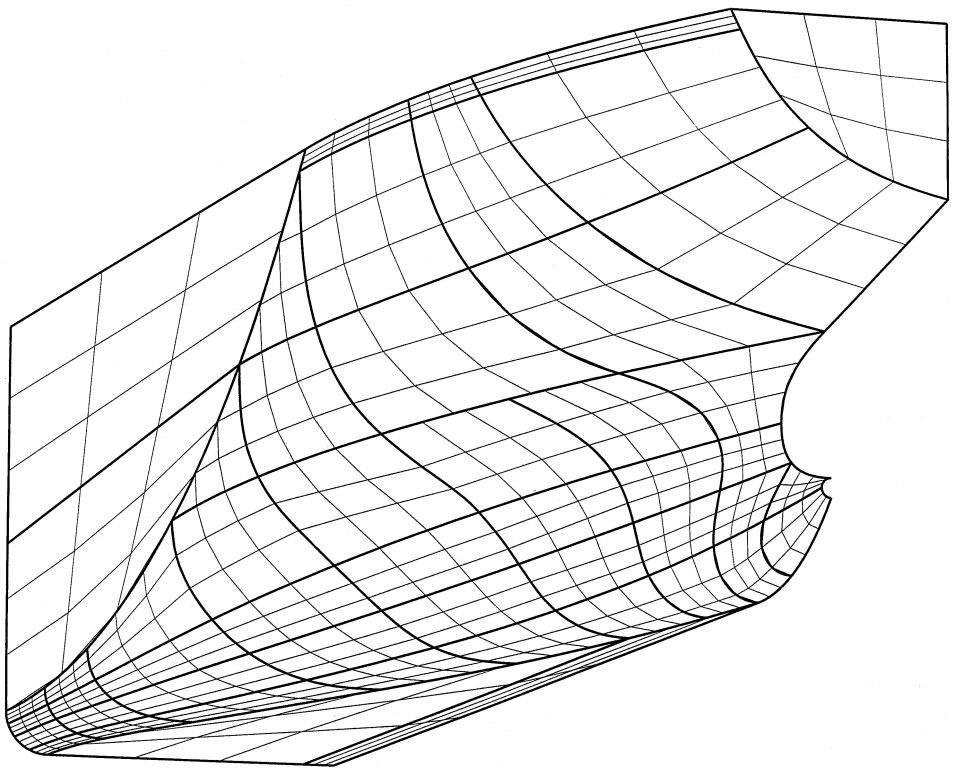


Figure 46 - Ship curves

The data associated with a Ship_curve are the following:

— curve_class;

— curve_shape;

— side_condition.

4.2.62.1 curve_class

The curve_class specifies the naval architectural categories for the Ship_curve that are commonly required for the design definition of Moulded_forms.

The curve_class shall be one of the following:

— bounding_line,

— buttock_line,

— camber,

— centreline,

— flat_of_bottom,

— flat_of_side,

— intersection_line,

— rise_of_floor,

— sheer,

— station_line,

— trace_line,

— unspecified,

— waterline.

4.2.62.1.1 buttock_line

buttock_line: a curve lying on the moulded surface of a hull that is the intersection of a longitudinal plane with the hull moulded form (see Figure 47).

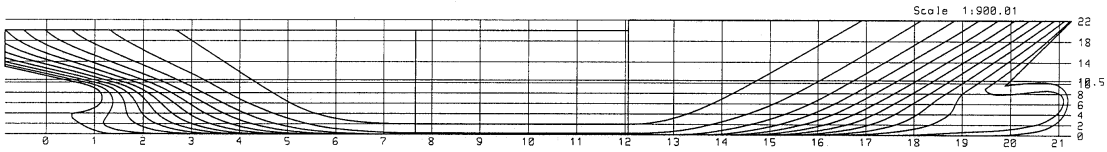


Figure 47 - Buttock lines

4.2.62.1.2 camber

camber: a curve lying in the moulded surface of a deck. It is generated by the intersection of a transverse plane with the deck moulded form.

4.2.62.1.3 centreline

centreline: a curve lying in the moulded surface of a hull. It is generated by the intersection of the longitudinal centreplane with the hull moulded form.

4.2.62.1.4 flat_of_bottom

flat_of_bottom: the boundary curve of the planar surface of the bottom at the base of a hull moulded form.

4.2.62.1.5 flat_of_side

flat_of_side: the boundary curve of the planar surface at the outer-most port or starboard side of the hull moulded form.

4.2.62.1.6 intersection_line

intersection_line: a curve that is generated by the intersection of two moulded_forms or two surfaces.

4.2.62.1.7 sheer

sheer: a curve lying in the moulded surface of a deck. It is generated by the intersection of a longitudinal plane with the deck moulded form.

4.2.62.1.8 station_line

station_line: a curve lying in the moulded surface of a hull. It is generated by the intersection of a transverse plane with the hull moulded form (see Figure 48).

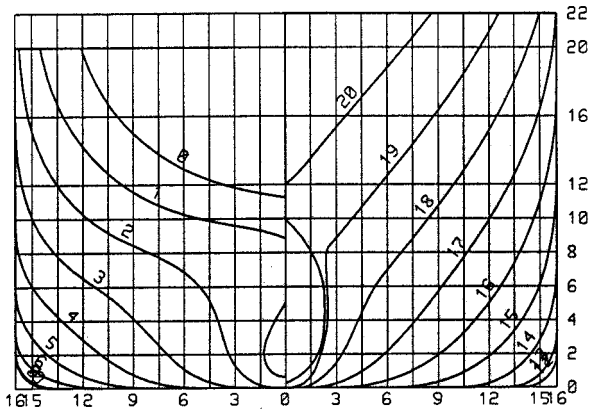


Figure 48 - Station lines

4.2.62.1.9 waterline

waterline: a curve lying in the moulded surface of a hull. It is generated by the intersection of the water plane with the hull moulded form (see Figure 49).

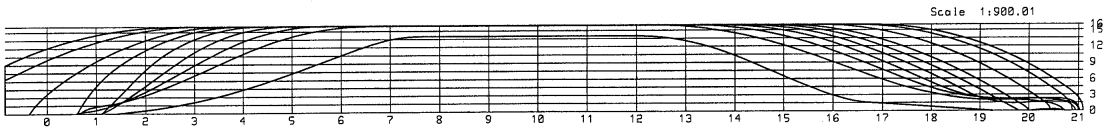


Figure 49 - Waterlines

4.2.62.1.10 bounding_line

bounding_line: a bounding curve of a moulded_form or a ship_surface.

4.2.62.1.11 rise_of_floor

rise_of_floor: the boundary curve of the rise of floor at the bottom of the hull moulded form where the bilge starts.

4.2.62.1.12 trace_line

trace_line: a curve in the moulded surface of a structural part, as its boundary or as the trace line for the attachment of a profile or girder.

4.2.62.1.13 unspecified

unspecified: a curve where the naval architectural meaning is not specified.

4.2.62.2 curve_shape

The curve_shape specifies the underlying geometric definition of the Ship_curve. See 4.3.93 for the application assertion.

4.2.62.3 side_condition

The side_condition specifies the behaviour of curves crossing the defined curve at the cross over point.

EXAMPLE - A section curve will knuckle at a waterline representing the intersection of the bilge keel and the flat of side.

The side_condition shall be one of the following:

— knuckle,

— smooth,

— tangent,

— unspecified.

4.2.62.3.1 knuckle

knuckle: the crossing curve will have a discontinuous tangent either side of the crossover point.

4.2.62.3.2 smooth

smooth: the crossing curve will be smooth.

4.2.62.3.3 tangent

tangent: the crossing curve will have a specified tangent.

4.2.62.3.4 unspecified

unspecified: for the crossing curve no tangency information is known.

4.2.63 Ship_curve_with_spacing_position

A Ship_curve_with_spacing_position is a type of Ship_curve (see 4.2.62) that has an associated spacing position.

EXAMPLE - A waterline that has the geometry defined by a B-spline curve and located at a vertical spacing position.

NOTE - A Ship_curve_with_spacing_position can only be defined for a 2 dimensional ship curve located in one of the planes of the global co-ordinate system.

The data associated with a Ship_curve_with_spacing_position is the following:

— location.

The location specifies Spacing_position where the ship curve is defined. See 4.3.95 for the application assertion.

4.2.64 Ship_surface

A Ship_surface is a type of Moulded_form_representation_item (see 4.2.12) that represents a Surface with associated naval architectural information.

The data associated with a Ship_surface are the following:

— surface_class;

— surface_shape.

4.2.64.1 surface_class

The surface_class specifies the naval architectural categories for the Ship_surface based on the location of the surface.

The surface_class shall be one of the following:

— blending_surface,

— external_surface,

— internal_surface.

4.2.64.1.1 blending_surface

blending_surface: a surface that forms a connection between Moulded_forms e.g., a hull and an appendage to the hull (thruster)

4.2.64.1.2 external_surface

external_surface: a surface that represents a Moulded_form or a part of it that is part of the outer surface representation of the ship

4.2.64.1.3 internal_surface

internal_surface: a surface that represents a Moulded_form or a part of it that is part of the inner surface representation of the ship e.g., girders or bulkheads

4.2.64.2 surface_shape

The surface_shape specifies the underlying geometric definition of the Ship_surface. See 4.3.103 for the application assertion.

IX UoF location_concepts

4.2.65 Global_axis_placement

A Global_axis_placement is a type of General_characteristics_definition (see 4.2.92) that defines a fixed system of right handed orthogonal axes to which geometric data are referred. A Global_axis_placement shall have a:

— positive z-axis in an upwards direction starting from the base of the ship,

— positive x-axis running along the ship on the intersection of the centreline with the base. In one case it is directed from the after part of the ship to the forward part of the ship, in the other it is directed from the forward part of the ship to the after part of the ship,

— origin of the global axis placement can be any point on the x axis.

The distance of the after perpendicular from the origin and the orientation of the x-axis shall be specified.

If any other system of axes is used, local or global, then the transformation relations between it and the Global_axis_placement shall be specified (see Figure 50).

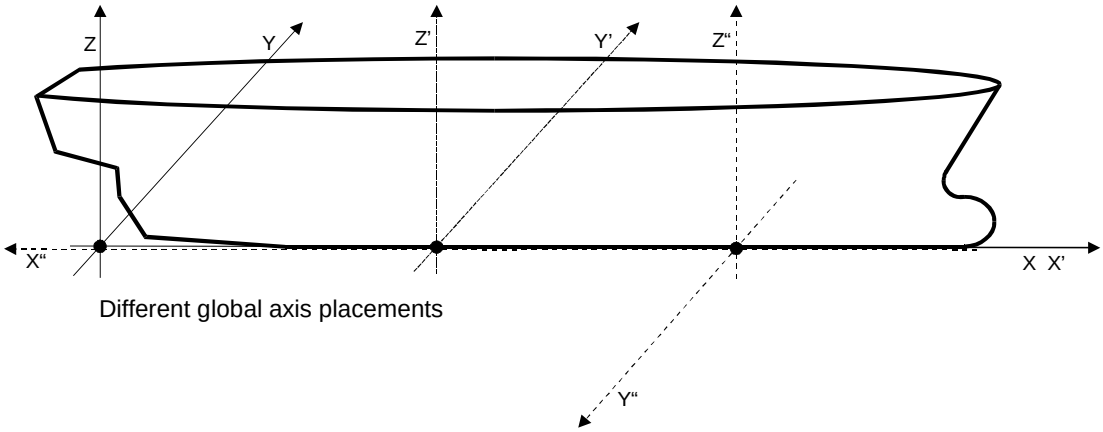


Figure 50 - Global axis placements

The data associated with a Global_axis_placement are the following:

— after_perpendicular_offset;

— orientation.

4.2.65.1 after_perpendicular_offset

The after_perpendicular_offset specifies the distance from the origin of the Global_axis_placement to the after perpendicular.

4.2.65.2 orientation

The orientation specifies the direction of the x-axis.

The orientation shall be one of the following:

— aft_pointing,

— forward_pointing.

4.2.65.2.1 aft_pointing

aft_pointing: an orientation of a right handed ship co-ordinate system that has the positive x axis from the forward part of the ship directed to the aft part of the ship.

4.2.65.2.2 forward_pointing

forward_pointing: an orientation of a right handed ship co-ordinate system that has the positive x axis from the aft part of the ship directed to the forward part of the ship.

4.2.66 Local_co_ordinate_system

A Local_co_ordinate_system is a type of Definition (see 4.2.88) and is a type of Axis2_placement_3d. Local_co-ordinate_system is used to establish a location in space. A Local_co_ordinate_system is always defined with respect to the surrounding Co_ordinate_system, the Global_axis_placement or another Local_co_ordinate_system (see Figure 51). A Local_co_ordinate_system may be a Local_co_ordinate_system_with_position_reference (see 4.2.67).

NOTE 1 - Axis2_placement_3d is a STEP resource entity defined in ISO 10303-42.

NOTE 2 - Local axes and origin are handled in the same way as for axis2_placement_3d.

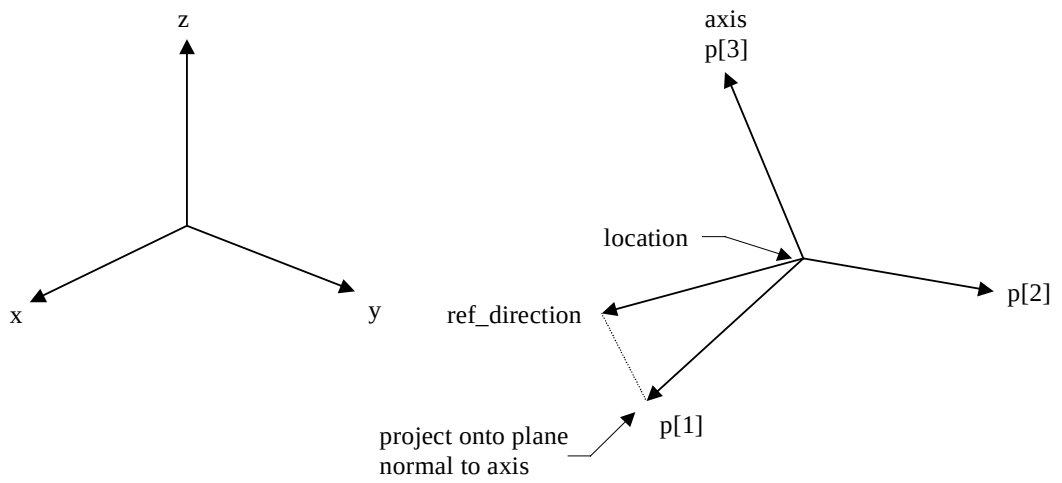


Figure 51 – Axis 2 placement 3d

NOTE 3 - A local_co_ordinate system will always form a right-handed system.

The data associated with a Local_co_ordinate_system is the following:

— parent.

The parent specifies the surrounding co-ordinate system, which serves as definition space for “this” coordinate system.

4.2.67 Local_co_ordinate_system_with_position_reference

A Local_co_ordinate_system_with_position_reference is a special type of Local_co_ordinate_system (see 4.2.66), which directly refers to the unique Global_axis_placement as its parent. Its location is defined by references to longitudinal, vertical or transversal frames, possibly using an additional offset value (a distance). Alternatively absolute co-ordinates may be specified. Combinations of co-ordinates and references are also valid. A Local_co_ordinate_system_with_position_reference shall be defined in the global system. The axes are required to be parallel to the axes of the Global_axis_placement.

The data associated with a Local_co_ordinate_system_with_position_reference are the following:

— location;

— longitudinal_ref;

— transversal_ref;

— vertical_ref.

4.2.67.1 location

The location specifies the origin of the local co-ordinate system defined in the parent Co_ordinate_system and derived from a local or possibly a global definition. See 4.3.64 for the application assertion.

4.2.67.2 longitudinal_ref

The longitudinal_ref is optional and, if present, refers to a Longitudinal_position possibly with an offset value or an absolute coordinate value along the longitudinal axis of the global co-ordinate system.

4.2.67.3 transversal_ref

The transversal_ref is optional and, if present, refers to a Transversal_position possibly with an offset value or an absolute co-ordinate value along the transversal axis of the global co-ordinate system.

4.2.67.4 vertical_ref

The vertical_ref is optional and, if present, refers to a Vertical_position possibly with an offset value or an absolute co-ordinate value along the vertical axis of the global co-ordinate system.

4.2.68 Spacing_table

A Spacing_table is a type of Definition (see 4.2.88), which is a collection of Spacing_positions that defines a list of reference points along one of the co-ordinate axes of the ship. There are several specific types of the Spacing_table, which can be chosen. A Spacing_table may be a Longitudinal_table (see 4.2.69), a Transversal_table (see 4.2.70) or a Buttock_table (see 4.2.72).

EXAMPLE - A frame table is a type of Spacing_table, where the frame positions would be longitudinal positions.

The data associated with a Spacing_table are the following:

— description;

— name;

— spacing_table_representations.

4.2.68.1 description

The description is optional and, if present, specifies the textual account of the reason why the Spacing_table was created and any additional text that is required to describe the purpose of the Spacing_table.

4.2.68.2 name

The name is optional and, if present, specifies the context specific identification for the Spacing_table.

4.2.68.3 spacing_table_representations

The spacing_table_representations specify the positions, which make up the table on the co-ordinate axis that are of interest. See 4.3.107 for the application assertion.

4.2.69 Longitudinal_table

A Longitudinal_table is a type of Spacing_table (see 4.2.68) that is a spacing table whose positions lie on the longitudinal axis of the global co-ordinate system, which is the global x-axis. A Longitudinal_table may be a Frame_table (see 4.2.73) or a Station_table (see 4.2.74).

The data associated with a Longitudinal_table is the following:

— spacing_table_representations.

The spacing_table_representations specify the longitudinal positions, which make up the longitudinal table. See 4.3.65 for the application assertion.

4.2.70 Transversal_table

A Transversal_table is a type of Spacing_table (see 4.2.68) that is a spacing table whose positions lie on the transverse axis of the global co-ordinate system, which is the global y-axis. A Transversal_table may be a Buttock_table (see 4.2.72).

The data associated with a Transversal_table is the following:

— spacing_table_representations.

The spacing_table_representations specify the transversal positions, which make up the transversal table. See 4.3.119 for the application assertion.

4.2.71 Vertical_table

A Vertical_table is a type of Spacing_table (see 4.2.68) that is a spacing table whose positions lie on the vertical axis of the global co-ordinate system, which is the global z-axis. A Vertical_table may be a Waterline_table (see 4.2.75).

The data associated with a Vertical_table is the following:

— spacing_table_representations.

The spacing_table_representations specify the vertical positions, which make up the vertical table. See 4.3.125 for the application assertion.

4.2.72 Buttock_table

A Buttock_table is a type of Transversal_table (see 4.2.70) that is a spacing table whose positions are a reference for the location of buttocks and are located on the global y axis.

4.2.73 Frame_table

A Frame_table is a type of Longitudinal_table (see 4.2.69) that is a spacing table whose positions are a reference for the location of frames and are located on the global x axis.

NOTE - Frames are used for the internal structure of the ship and they are structural elements. A ship can have more than 100 frames. The intersection curve between a frame and the hull moulded form is a curve of transversal section through the ship hull.

4.2.74 Station_table

A Station_table is a type of Longitudinal_table (see 4.2.69) that is a type of transversal table whose positions are a reference for the location of stations and are located on the global x axis.

NOTE - Stations are used in the design process of a ship and the station curves are curves of transversal sections through the ship hull. There are usually 20 stations, but the number can differ from shipyard to shipyard.

4.2.75 Waterline_table

A Waterline_table is a type of vertical table (see 4.2.71) that is a spacing table whose positions are a reference for the location of waterlines and are located on the global z axis.

4.2.76 Spacing_position

A Spacing_position is a position on one of the global co-ordinate axes of the ship that is used as a reference point for any geometrical or structural item during the design and manufacture of the ship (see Figure 52). A Spacing_position may be a Spacing_position_with_offset (see 4.2.77), a Longitudinal_position (see 4.2.78), a Transversal_position (see 4.2.79) or a Vertical_position (see 4.2.80).

EXAMPLE - Typically spacing positions are specified by LFR 123, TFR 10, 100, 100.1, A. In addition the distance to the global origin is defined for instance by 154.5 meters.

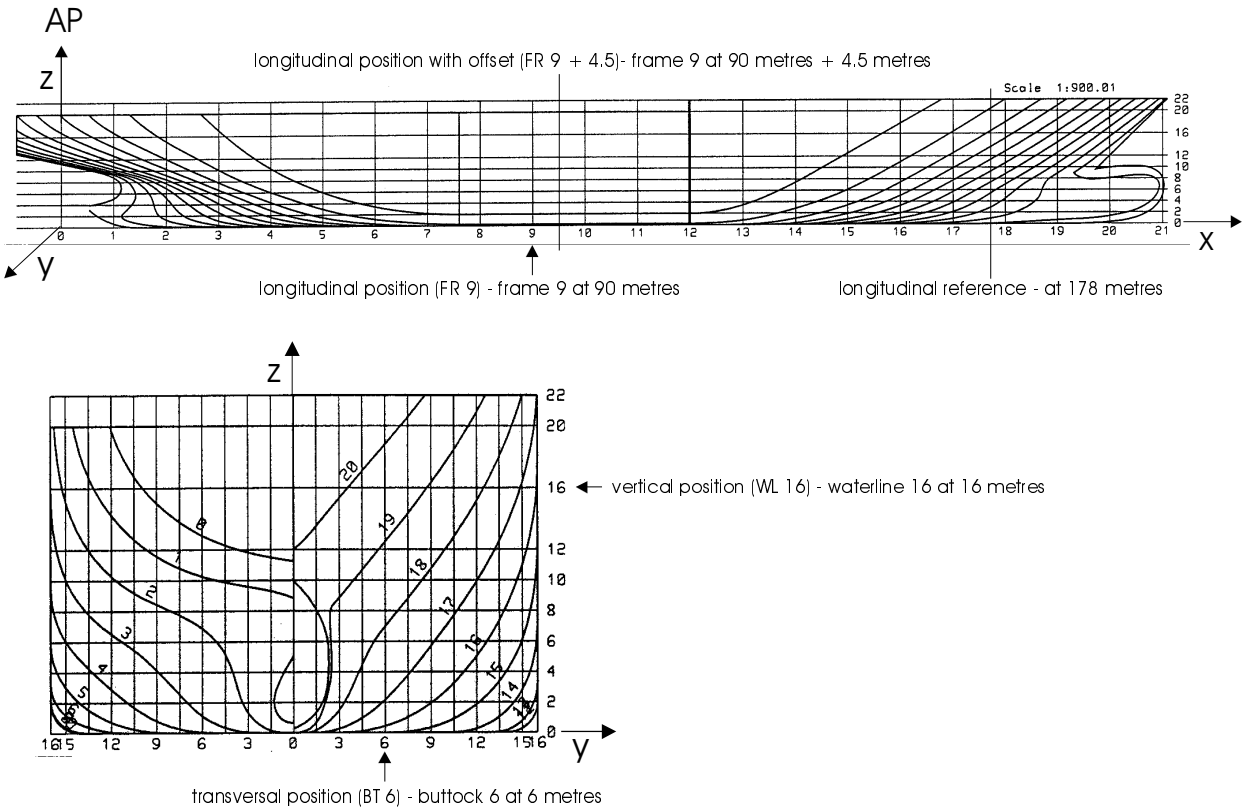


Figure 52 – Spacing position

The data associated with a Spacing_position are the following:

— name;

— position;

— position_number.

4.2.76.1 name

The name is optional and, if present, specifies a label that is used to name the reference point. This is used where the spacing positions are identified by non-numeric names, which cannot use the REAL identifier.

4.2.76.2 position

The position specifies the distance to the origin of the ships global co-ordinate system. The axis on which the distance is measured depends on the type of Spacing_position.

4.2.76.3 position_number

The position_number specifies the numerical identification, which is given to the Spacing_position.

4.2.77 Spacing_position_with_offset

A Spacing_position_with_offset is a type of Spacing_position (see 4.2.76) that is a position defined by an offset to an existing spacing position on one of the global co-ordinate axes of the ship. It is used as a reference point for any geometrical or structural item during the design and manufacture of the ship.

The data associated with a Spacing_position_with_offset are the following:

— offset;

— position;

— relating_spacing_position.

4.2.77.1 offset

The offset specifies the distance to the relating spacing position. The axis on which the distance is measured depends on the type of the relating Spacing_position.

4.2.77.2 position

The position specifies the distance to the origin of the ships global co-ordinate system. The axis on which the distance is measured depends on the type of the relating Spacing_position.

4.2.77.3 relating_spacing_position

The relating_spacing_position specifies the spacing position from where the offset is taken to identify the Spacing_position_with_offset. See 4.3.106 for the application assertion.

4.2.78 Longitudinal_position

A Longitudinal_position is a type of Spacing_position (see 4.2.76) that is located on the global x-axis.

4.2.79 Transversal_position

A Transversal_position is a type of Spacing_position (see 4.2.76) that it is located on the global y-axis.

4.2.80 Vertical_position

A Vertical_position is a type of Spacing_position (see 4.2.76) that is located on the global z-axis.

X UoF items

4.2.81 Definable_object

A Definable_object is an abstract concept (SUPERTYPE) of a placeholder for any physical object in the ship, where then life cycle dependent definitions can point to. A Definable_object may be an Item (see 4.2.83), an Item_relationship (see 4.2.84) or an Item_structure (see 4.2.85).

The data associated with a Definable_object are the following:

— definitions;

— id.

4.2.81.1 definitions

The definitions specify the definitions pointing to the Definable_object. See 4.3.20 for the application assertion.

4.2.81.2 id

The id specifies the global unique identifier for the Definable_object. See 4.3.21 for the application assertion.

4.2.82 Versionable_object

A Versionable_object is an abstract concept (SUPERTYPE) for any physical object in the ship or any definition that can carry versioning information. A Versionable_object may be a Definition (see 4.2.88) , a Document (see 4.2.112) an Item_relationship (see 4.2.84) or an Item_structure (see 4.2.85).

The data associated with a Versionable_object is the following:

— version_id.

The version_id specifies the version identifier of the Versionable_object.

4.2.83 Item

An Item is a type of Definable_object (see 4.2.81) that is a discrete, identifiable thing used in one or more production activities. An Item is something (to be) created by a physical or mental activity or (automatically) derived from one or more other Item s. An Item needs not to represent a physical thing. It may also represent some abstract concept like "activity", "task", etc. Item provides the functionality to have relationships to other Items and to be member in an Item_structure. Each entity that is intended to have such functionality should inherit from Item. An Item may be a Ship (see 4.2.86), a Ship_moulded_form (see 4.2.1) or a Moulded_form (see 4.2.2).

The data associated with an Item are the following:

— description;

— documentation;

— name;

— ship_context.

4.2.83.1 description

The description is optional and, if present, specifies the description for the Item.

4.2.83.2 documentation

The documentation specifies additional documents available (if it exists) for the Item. See 4.3.52 for the application assertion.

4.2.83.3 name

The name specifies the (human readable) name of the concept that is represented by the Item.

4.2.83.4 ship_context

The ship_context is optional and, if present, specifies the context of the Item in terms of its applicability to a ship. It establishes the link to ship, the root entity in the ship product model. See 4.3.53 for the application assertion.

4.2.84 Item_relationship

An Item_relationship is a type of Definable_object (see 4.2.81) and is a type of Versionable_object (see 4.2.82). An Item_relationship defines the association of two Items. This means the related Items share common things or activities or are dependent from each other. Each entity that is intended to relate Items in any way should inherit from Item_relationship. An Item_relationship may be a Moulded_form_relationship (see 4.2.3).

The data associated with an Item_relationship are the following:

— contxt.

— external_item_1;

— external_item_2;

— item_1;

— item_2;

4.2.84.1 contxt

The contxt is optional and, if present, describes additional information about the relationship and its items.

4.2.84.2 external_item_1

The external_item_1 is optional and, if present, specifies the relating Item in the Item_relationship that is defined in a different exchange file and is accessed through an external instance reference. Either item_1 or external_item_1 have to be present in the Item_relationship. See 4.3.54 for the application asseertion.

4.2.84.3 external_item_2

The external_item_2 is optional and, if present, specifies the related Item in the Item_relationship that is defined in a different exchange file and is accessed through an external instance reference. Either

item_2 or external_item_2 have to be present in the Item_relationship. See 4.3.55 for the application asseertion.

4.2.84.4 item_1

The item_1 is optional and, if present, specifies the relating Item in the Item_relationship that is defined in the same exchange file. See 4.3.56 for the application asseertion.

4.2.84.5 item_2

The item_2 is optional and, if present, specifies the related Item in the Item_relationship that is defined in the same exchange file. See 4.3.57 for the application asseertion.

4.2.85 Item_structure

An Item_structure is a type of Definable_object (see 4.2.81) and is a type of Versionable_object (see 4.2.82). An Item_structure represents a container able to hold Items possibly related by Item_relationships. An Item_structure forms a graph without any restrictions regarding the number of entries or cycles and the connectivity. Each entity that is intended to collect Items and their relationships in a specified context should inherit from Item_structure. An Item_structure may be a Ship_moulded_form (see 4.2.1).

The data associated with an Item_structure are the following:

— external_items;

— external_relationships;

— items;‘

— relationships.

4.2.85.1 external_items

The external_items specify a collection of Item_relationships in the Item_structure that are defined in a different exchange file and are accessed through external instance reference. See 4.3.58 for the application assertion.

4.2.85.2 external_relationships

The external_relationships specify a collection of Item_relationships in the Item_structure that are defined in a different exchange file and are accessed through external instance reference. See 4.3.59 for the application assertion.

4.2.85.3 items

The items specify a collection of Items in the Item_structure that are defined in the same exchange file.

The external_items specify a collection of Item_relationships in the Item_structure that are defined in a different exchange file and are accessed through external instance reference. See 4.3.60 for the application assertion.

4.2.85.4 relationships

The relationships specify a collection of Item_relationships in the Item_structure that are defined in the same exchange file. See 4.3.61 for the application assertion.

4.2.86 Ship

A Ship is a type of Item (see 4.2.83) that represents the root entity in the ship product model where all data defining the product ship are related too. It exists in any life cycle stage. A project, which represents a ship in the early design phase, for example would only include a definition with some basic parameters pointing to the ship.

The data associated with a Ship are the following:

— ship_items;

— units.

4.2.86.1 ship_items

The ship_items specify the Items that reference the Ship. See 4.3.92 for the application assertion.

4.2.86.2 units

The units specify a reference to a set of pre-defined units for all types of measures that may appear in the ship model.

NOTE - The name of the ship is specified as a Definition in Ship_designation, where it may be versioned.

4.2.87 Global_id

The Global_id is a persistent, global identifier which uniqely identifies a Definable_object or a definition.

The data associated with a Global_id are the following:

— company_id;

— id;

— local_id.

4.2.87.1 company_id

The company_id specifies a unique identifier for the company, which created the data. The string is left justified and blank filled.

4.2.87.2 id

The id specifies a unique identifier, which is a combination of company_id and local_id, which are seperated by a dot.

4.2.87.3 local_id

The local_id specifies a persistent identifier, which uniquely identifies this definition throughout the company. Assigned at the time a definition is created. The string is left justified and blank filled.

XI UoF definitions

4.2.88 Definition

Definition is a type of Versionable_object (see 4.2.82) that represents an abstract concept (SUPERTYPE) for all life cycle dependent definitions. A Definition may be a Design_definition (see 4.2.90), a Functional_definition (see 4.2.91), a General_characteristics_definition (see 4.2.92), a Moulded_form_characteristics_definition (see 4.2.27), Ship_overall_dimensions (see 4.2.26), a Spacing_table (see 4.2.68) or a Local_co_ordinate_system (see 4.2.66).

The data associated with a Definition are the following:

— defined_for;

— id;

— local_units.

4.2.88.1 defined_for

The defined_for specifies the Definable_objects, which are defined by Definition. See 4.3.22 for the application assertion.

4.2.88.2 id

The id specifies the global unique identifier for the definition. See 4.3.23 for the application assertion.

4.2.88.3 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship.

4.2.89 Definition_relationship

The Definition_relationship provides the possibility of relating two Definitions that share common things or activities or are dependent from each other.

The data associated with a Definition_relationship are the following:

— definition_1;

— definition_2;

— description.

4.2.89.1 definition_1

The definition_1 specifies the related definition in the Definition_relationship. See 4.3.29 for the application assertion.

4.2.89.2 definition_2

The definition_2 specifies the relating definition in the Definition_relationship. See 4.3.30 for the application assertion.

4.2.89.3 description

The description is optional and, if present, specifies additional information about the relationship.

4.2.90 Design_definition

The Design_definition is a type of Definition (see 4.2.88) that represents an abstract concept (SUPERTYPE) for all design definitions. A Design_definition may be a Moulded_form_design_definition (see 4.2.4), a Hydrostatic_definition (see 4.2.38) or a Stability_definition (see 4.2.49).

The data associated with a Design_definition is the following:

— representations.

The representations specify the representations of the design definition. It is possible for a Design_definition to have multiple Representations. See 4.3.30 for the application assertion.4.3.31.

4.2.91 Functional_definition

The Functional_definition is a type of Definition (see 4.2.88) that represents an abstract concept (SUPERTYPE) for all functional definitions. It specifies the function of a Definable_object. A Functional_definition may be aMoulded_form_functional_definition (see 4.2.5) or a Ship_type (see 4.2.13).

The data associated with a Functional_definition is the following:

— user_def_function.

The user_def_function is optional and, if present, specifies a function for a Definable_object provided by the user.

4.2.92 General_characteristics_definition

The General_characteristics_definition is a type of Definition (see 4.2.88) that provides a major part of the characterisation of the ship. It includes information about the ship type, main dimensions, capacities, owner, shipyard and classification of the ship. A Genearl_characteristics_definition may be a Class_and_statutory_designation (see 4.2.20), a Owner_designation (see 4.2.25), a Ship_designation (see 4.2.23), a Shipyard_designation (see 4.2.24), Principal_characteristics (see 4.2.19), Class_parameters (see 4.2.18) or a Global_axis_placement (see 4.2.65).

The data associated with a General_characteristics_definition is the following:

— defined_for.

The defined_for specifies the Ship or set of Ships for which the General_characteristics_definition applies. See 4.3.40 for the application assertion.

XII UoF configuration_management

4.2.93 Approval_event

An Approval_event is a type of Event (see 4.2.107) that specifies an event, which follows in a change of the Approval_status.

The data associated with an Approval_event are the following:

— approval_reference;

— result;

— user_defined_result.

4.2.93.1 approval_reference

The approval_reference specifies the Approval_history affected by the event. Every Approval_event must refer to exactly one Approval_history. See 4.3.1 for the application assertion.

4.2.93.2 result

The result specifies the Approval_status the event leads to.

The result shall be one of the following:

— approved,

— rejected,

— unapproved,

— user_defined.

4.2.93.2.1 approved

approved: the definition is approved.

4.2.93.2.2 rejected

rejected: the definition is rejected and has to be redeveloped.

4.2.93.2.3 unapproved

unapproved: a definition has not yet been approved.

4.2.93.2.4 user_defined

user_defined: the approval result is user defined.

4.2.93.3 user_defined_result

The user_defined_result is optional and, if present, specifies a user-defined approval status. The attribute shall only be given when Approval_status.result is USER_DEFINED.

4.2.94 Approval_history

An Approval_history collects all Approval_events of a specific type for a Definition.

The data associated with an Approval_history are the following:

— approvals;

— status;

— subject.

4.2.94.1 approvals

The approvals specify the sequence of Approval_events having occurred up to this point in time. Every Approval must have an associated Approval_status, thus the history must consist of at least one Approval_event. The list is assumed to build up in chronological order. See 4.3.2 for the application assertion.

4.2.94.2 status

The status specifies the current approval status.

The status shall be one of the following:

— approved,

— rejected,

— unapproved,

— user_defined.

4.2.94.2.1 approved

approved: the definition is approved.

4.2.94.2.2 rejected

rejected: the definition is rejected and has to be redeveloped.

4.2.94.2.3 unapproved

unapproved: a definition has not yet been approved.

4.2.94.2.4 user_defined

user_defined: the approval result is user defined

4.2.94.3 subject

The subject specifies the Definition as to where the Approval_history is related. See 4.3.2 for the application assertion.

NOTE - A Definition may have zero, one or many associated Approvals. In case it has more then one associated Approval, all of them shall be different.

4.2.95 Change

A Change represents a change affecting one or more definitions. The Change is linked to the related Change_state.

The data associated with a Change are the following:

— class;

— possible_states.

4.2.95.1 class

The class specifies additional information about the role of the change.

EXAMPLE - Headquater Modification Request, Engineering Change Proposal or such possible qualifications.

4.2.95.2 possible_states

The possible_states links a Change with its associated Change_states. See 4.3.4 for the application assertion.

4.2.96 Change_state

A Change_state is an abstract concept (SUPERTYPE) for the specific stages of a Change. A Change_state may be a Change_plan (see 4.2.100), aChange_realization (see 4.2.101) or a Change_request (see 4.2.97).

The data associated with a Change_state are the following:

— author;

— change_reference;

— date_time.

4.2.96.1 author

The author specifies the person and / or organization responsible for the Change_state activities during the period, lasting from the end of the previous (if it exists) up to the end of this Change_state. See 4.3.15 for the application assertion.

4.2.96.2 change_reference

The change_reference specifies the Change, which the data of the Change_state is referring to. See 4.3.13 for the application assertion.

4.2.96.3 date_time

The date_time specifies the date and time when the Change_state was (approximately) reached. See 4.3.14 for the application assertion.

4.2.97 Change_request

A Change_request is a type of Change_state (see 4.2.96) that represents the first phase of a Change, where the needs for a Change and possible solution_alternatives are established.

The data associated with a Change_request are the following:

— addressee;

— initiator;

— problem;

— solution_alternatives;

— solution_description.

4.2.97.1 addressee

The addressee is optional and if present, specifies the person and/ or organization where the request is addressed to. See 4.3.11 for the application assertion.

4.2.97.2 initiator

The initiator specifies the person and/ or organization where the request is coming from. See 4.3.12 for the application assertion.

4.2.97.3 problem

The problem specifies a textual description of the problem responsible for the request.

4.2.97.4 solution_alternatives

The solution_alternatives specify alternative solutions envisaged to solve the problem. A solution is described in terms of the effect on the Definitions, and associated costs. See 4.3.10 for the application assertion.

4.2.97.5 solution_description

The solution_description is optional and if present, specifies a textual description of one or more possible solutions for the problem. This textual description should be present if the solution_alternatives are not yet established, or may enhance the information provided by the solution_alternatives.

4.2.98 Check

A Check is a type of Event (see 4.2.107) that defines the details of a planned or fulfilled control activity.

4.2.99 Change_impact

A Change_impact defines the effect a Change will cause or has caused.

The data associated with a Change_impact is the following:

— impact.

The impact specifies the effect of a Change in terms of the creation, modification or deletion of some Definitions. See 4.3.5 for the application assertion.

4.2.100 Change_plan

A Change_plan is a typ of Change_state (see 4.2.96) that represents the proposed solution for a Change. It is the basis for the activities necessary to actually realize the Change.

The data associated with a Change_plan are the following:

— checks;

— impact.

4.2.100.1 checks

The checks specify the checks planned for the Change. See 4.3.7 for the application assertion.

4.2.100.2 impact

The impact specifies the estimated or calculated effects of the Change. This impact is usually chosen from the set of change_request.solution_alternatives. See 4.3.6 for the application assertion.

4.2.101 Change_realization

A Change_realization is atype of Change_state (see 4.2.96) that defines the activities to realise the Change.

The data associated with a Change_realization are the following:

— checks;

— impact.

4.2.101.1 checks

The checks specify the checks performed to realise the Change. See 4.3.9 for the application assertion.

4.2.101.2 impact

The impact specifies the impact upon definitions that the realisation of the Change will have.

4.2.102 Definition_change_event

A Definition_change_event is a type of Event (see 4.2.107) the generalization of the events effectively change a Definition. A Definition_change_event may be aEnvisaged_definition_creation (see 4.2.103), a Definition_modification (see 4.2.105), a Definition_creation (see 4.2.104) or a Definition_deletion (see 4.2.106). See 4.3.8 for the application assertion.

4.2.103 Envisaged_definition_creation

An Envisaged_definition_creation is a type of Definition_change_event (see 4.2.102) that represents the event leading to a new Definition. The event is an envisaged event and has not yet happened. The Definition as the subject of the event does not yet exist and is described in terms of descriptive, non-formal properties.

The data associated with an Envisaged_definition_creation are the following:

— base;

— definition_category;

— definition_description.

4.2.103.1 base

The base specifies the Definitions the envisaged definition is derived from. See 4.3.35 for the application assertion.

4.2.103.2 definition_category

The definition_category specifies the category the envisaged definition belongs to.

4.2.103.3 definition_description

The definition_description specifies a textual description of the significant features of the envisaged definition.

4.2.104 Definition_creation

A Definition_creation is a type of Definition_change_event (see 4.2.102) that represents the event leading to a new Definition. The base Definition shall be populated if the subject Definition is based upon an existing Definition. The base Definition need not be the immediately preceding version of the subject Definition. However, it may refer to any previous version in the version history of the same Item, or to any Definition of another Item which contributes to the creation of the subject Definition.

The data associated with a Definition_creation are the following:

— base;

— subject.

4.2.104.1 base

The base specifies the Definitions the subject is derived from. See 4.3.24 for the application assertion.

4.2.104.2 subject

The subject specifies the Definition created by the change event. See 4.3.25 for the application assertion

4.2.105 Definition_modification

A Definition_modification is a type of Definition_change_event (see 4.2.102) that represents the event leading to a change of a Definition. The base Definition need not be the immediately preceding version of the subject Definition, but may refer to any previous version in the version history of the same Item.

The data associated with a Definition_modification are the following:

— base;

— subject.

4.2.105.1 base

The base specifies the Definitions the subject is derived from. See 4.3.27 for the application assertion.

4.2.105.2 subject

The subject specifies the Definition modified or to be modified by the change event. See 4.3.28 for the application assertion.

4.2.106 Definition_deletion

A Definition_deletion is a type of Definition_change_event (see 4.2.102) that represents the event leading to the deletion of a Definition.

The data associated with a Definition_deletion is the following:

— subject.

The subject specifies the Definition deleted or to be deleted by the change event. See 4.3.26 for the application assertion.

4.2.107 Event

An Event identifies that something has happened at a certain time, activated by a certain person for a certain reason. An Event may be a Check (see 4.2.98), aDefinition_change_event (see 4.2.102) or an Approval_event (see 4.2.93).

The data associated with an Event are the following:

— caused_by;

— caused_when;

— description.

4.2.107.1 caused_by

The caused_by specifies the person creating the Event. See 4.3.37 for the application assertion.

4.2.107.2 caused_when

The caused_when specifies the date and time, the Event occurred. See 4.3.36 for the application assertion.

4.2.107.3 description

The description specifies additional textural information for the Event.

4.2.108 Revision

The Revision specifies a collection of Versionable_objects (definitions, documents, item relationships and item structures) which build a revision at a specific stage in the life cycle of the ship. Each Versionable_object is only used with one specific version in the revision. A revision is not created automatically, but has to be created explicitly each time it is needed. A Revision may be a Revision_with_context (see 4.2.109).

The data associated with a Revision are the following:

— members;

— name;

— reason.

4.2.108.1 members

The members specify the Versionable_objects the Revision consists of. See 4.3.86 for the application assertion.

4.2.108.2 name

The name specifies a Label which identifies a particular revision.

4.2.108.3 reason

The reason specifies a textual description of what caused the creation of a new revision.

4.2.109 Revision_with_context

The Revision_with_context is a type of Revision (see 4.2.108) that represents a collection of Versionable_objects (definitions, documents, item relationships and item structures) which build this revision at a specific stage in the life cycle of the ship. Each Versionable_object is only used with one specific version in the revision. In addition, the Revision_with_context provides a link to a Definable_object, which gets defined by the revision. The Revision_with_context is not created automatically, but has to be created explicitly each time it is needed. The Revision_with_context may be a Ship_moulded_form_revision (see 4.2.6).

The data associated with a Revision_with_context is the following:

— context_of_revision.

The context_of_revision specifies the link to a Definable_object, which gets defined by the revision.

4.2.110 Version_history

A Version_history identifies Versionable_objects and their Version_relationships in terms of their role as predecessors, respectively successors and with respect to each other. The Version_history shall be a directed acyclic graph. Consequently the Version_history may contain Versionable_objects considered alternatives with respect to each other (a Versionable_object having more than one successor), and merged Versionable_objects (a Versionable_object having more than one predecessor).

The data associated with a Version_history are the following:

— current_version;

— relationships;

— versions.

4.2.110.1 current_version

The current_version specifies the Versionable_object, which plays the role of the Current_version in this Version_history. See 4.3.121 for the application assertion.

4.2.110.2 relationships

The relationships specify the Version_relationships the Version_history consists of. See 4.3.120 for the application assertion.

4.2.110.3 versions

The versions specify the Versionable_objects the Version_history consists of. See 4.3.122 for the application assertion.

4.2.111 Version_relationship

A Version_relationship defines the relationship of two Versionable_objects of same type in terms of a Version_history.

The data associated with a Version_relationship are the following:

— predecessor;

— reason;

— successor.

4.2.111.1 predecessor

The predecessor specifies the Version the successor is derived from. See 4.3.123 for the application assertion.

4.2.111.2 reason

The reason specifies why a certain person created the new version, at a certain time.

4.2.111.3 successor

The successor specifies the Version the predecessor is the preceding version of. See 4.3.124 for the application assertion.

XIII UoF external_references

4.2.112 Document

A Document is a type of Versionable_object (see 4.2.82) that represents an unambiguous identification of some human readable data item defined outside ISO 10303. A document has an author, a title and may be versioned.

The data associated with a Document are the following:

— document_subset;

— has_author;

— has_title;

— source_type;

— summary.

4.2.112.1 document_subset

The document_subset lists zero or many subsets of the document. See 4.3.33 for the application assertion.

4.2.112.2 has_author

The has_author specifies the person and/ or organisation that authored the Document.

4.2.112.3 has_title

The has_title specifies a descriptive title consisting of a word, or group of words.

4.2.112.4 source_type

The source_type specifies the type of the document, whether it is a printed document, or an electronic document including the Type (PDF, Word, …).

4.2.112.5 summary

The summary is optional and if present, specifies a short textual description of the content of a document.

4.2.113 Document_usage_constraint

A Document_usage_constraint defines the applicability of a Document. Applicability may be defined in terms of selecting a specific part of a document and/ or interpreting the content of that specific part.

The data associated with a Document_usage_constraint are the following:

— element_name;

— line_number;

— page;

— paragraph;

— section;

— source.

4.2.113.1 element_name

The element_name specifies the name for this subset of the document.

4.2.113.2 line_number

The line_number is optional and if present, specifies a line number.

4.2.113.3 page

The page is optional and if present, specifies a page number. This may not always be a plain number, but also a roman number or a combination of chapter and page number.

4.2.113.4 paragraph

The paragraph is optional and if present, specifies a paragraph identifier.

4.2.113.5 section

The section is optional and if present, specifies a section label.

4.2.113.6 source

The source specifies to which document the following specification of sections, pages or paragraphs are related. See 4.3.34 for the application assertion.

4.2.114 Document_reference

A Document_reference defines the qualification of a Document in terms of its source or location. If the source is, i.e. a book, the pointer could be a section label or a page number. A Document_reference may be a Document_reference_with_address (see 4.2.115).

The data associated with a Document_reference is the following:

— assigned_document.

The assigned_document specifies the document (or portion of a document) which is to be associated with the product data.

4.2.115 Document_reference_with_address

A Document_reference_with_address is a type of Document_reference (see 4.2.114) and is a type of External_reference (see 4.2.117) that represents an External_reference to a document with a pointer to a location inside the document.

4.2.116 Universal_resource_locator

The address of an electronic data source (i.e. an internet address) This is an alternative to the common mail address as provided by ISO 10303 –41.

The data associated with an Universal_resource_locator are the following:

— location;

— machine_adress;

— other_protocol_type;

— port;

— protocol.

—

4.2.116.1 location

The location specifies the path on the target machine where the document is located.

4.2.116.2 machine_adress

The machine_adress specifies the name of the target machine that provides the service (i.e. the ftp server name or an internet address).

4.2.116.3 other_protocol_type

The other_protocol_type is optional and if present, specifies a transmission protocol apart from the ones used in Protocol_type.

4.2.116.4 port

The port is optional and if present, specifies for some protocols (i.e. http) another port number then the standard one.

4.2.116.5 protocol

The protocol specifies the type of the electronic transmission protocol (i.e. ftp, http...). Because there is a permanent development on this field it is not possible to cover all available protocol types. Due to this a protocol type USER_DEFINED is included and the optional other_protocol_type attribute is used to hold the protocol type in this case.

The protocol shall be one of the following:

— FTP,

— HTTP,

— USER_DEFINED.

4.2.116.5.1 FTP

FTP: file transfer protocol

4.2.116.5.2 HTTP

HTTP: hyper text transfer protocol

4.2.116.5.3 USER_DEFINED

USER_DEFINED: user defined transfer protocol

4.2.117 External_reference

An External_reference is a reference to an externally defined data source established by a link or through its address. An External_reference may be a Document_reference_with_address (see 4.2.115).

EXAMPLE - A WWW uniform resource locator denotes such a data source.

The data associated with an External_reference are the following:

— description;

— location.

4.2.117.1 description

The description specifies some additional information.

4.2.117.2 location

The location specifies the location of the externally defined data source e.g., a Universal_resource_locator for a document in the www.

4.2.118 External_instance_reference

An External_instance_reference represents a reference to an instance of an entity that does not exist in the same scope or STEP physical file.

The data associated with an External_instance_reference are the following:

— entity_type;

— schema_name;

— target_GUID;

— version.

4.2.118.1 entity_type

The entity_type specifies the name of the type of the externally referenced instance.

4.2.118.2 schema_name

The schema_name specifies the name of the schema where the type of the externally referenced instance is defined.

4.2.118.3 target_GUID

The target_GUID specifies the global unique identifier of the externally referenced instance. See 4.3.39 for the application assertion.

4.2.118.4 version

The version is optional and if present, specifies the version of the externally defined instance.

XIV UoF ship_measures

4.2.119 Extended_conversion_based_unit

An Extended_conversion_based_unit is a type of Named_unit that is based on an Extended_measure_with_unit. It is an extension of the ISO 10303-41 Entity Conversion_based_unit.

NOTE - Named_unit is a STEP resource entity defined in ISO 10303-41.

EXAMPLE - An inch is an Extended_conversion_based_unit. It is a unit from the Imperial system, its name is inch and it can be related to the Si_unit, millimetre, through an Extended_measure_with_unit whose value is 25.4 millimetre. A foot is also an Extended_conversion_based_unit. It is also a unit from the Imperial system, its name is foot and it can be related to the si unit, millimetre, either directly or through the unit called inch.

The data associated with an Extended_conversion_based_unit are the following:

— conversion_factor;

— name.

4.2.119.1 conversion_factor

The conversion_factor specifies the physical quantity from which the Extended_conversion_based_unit is derived. See 4.3.38 for the application assertion.

4.2.119.2 name

The name specifies the word or group of words by which the Extended_conversion_based_unit can be referenced.

4.2.120 Speed_unit

A Speed_unit is a type of Derived_unit that represents the unit of the physical quantity speed, which is distance made over time.

NOTE - Derived_unit is a STEP resource entity defined in ISO 10303-41.

4.2.121 Extended_measure_with_unit

An Extended_measure_with_unit is the specification of a physical quantity as defined in ISO 31 (clause 2). An Extended_measure_with_unit may be a Speed_measure_with_unit (see 4.2.122).

The data associated with an Extended_measure_with_unit are the following:

— unit_component;

— value_component.

4.2.121.1 unit_component

The unit_component specifies the unit of the physical quantity.

4.2.121.2 value_component

The value_component specifies the value of the physical quantity when expressed in the specified units.

4.2.122 Speed_measure_with_unit

A Speed_measure_with_unit is a type of Extended_measure_with_unit where the physical quantity is speed.

NOTE - Extended_measure_with_unit is a STEP resource entity defined in ISO 10303-41.

4.3 Application Assertions

This subclause specifies the application assertions for the ship moulded forms application protocol. Application assertions specify the relationships between application objects, the cardinality of the relationships and the rules required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

4.3.1 approval_event to approval_history

Each approval_event has approval_reference defined by exactly one approval_history. An approval_history defines the approval_reference for an approval_event.

4.3.2 approval_history to approval_event

Each approval_history has approvals defined by one up to many approval_event. A list of approval_event objects defines the approvals for an approval_history.

4.3.3 approval_history to definition

Each approval_history has subject defined by exactly one definition. A definition defines the subject for an approval_history.

4.3.4 change to change_state

Each change has possible_states defined by three, up to many, change_state. A set of change_state objects define the possible_states for a change.

4.3.5 change_impact to definition_change_event

Each change_impact has impact defined by one, up to many, definition_change_event. A set of definition_change_event objects definethe impact for a change_impact.

4.3.6 change_plan to change_impact

Each change_plan has impact defined by exactly one change_impact. A change_impact defines the impact for a change_plan.

4.3.7 change_plan to check

Each change_plan has checks defined by zero, up to many, check. A set of check objects defines the checks for a change_plan.

4.3.8 change_realization to change_impact

Each change_realization has impact defined by exactly one change_impact. A change_impact defines the impact for a change_realization.

4.3.9 change_realization to check

Each change_realization has checks defined by zero, up to many, check. A set of check objects defines the checks for a change_realization.

4.3.10 change_request to change_impact

Each change_request has solution_alternatives defined by zero, up to many, change_impact. A set of change_impact objects defines the solution_alternatives for a change_request.

4.3.11 change_request to person_and_organisation

Each change_request has initiator defined by exactly one person_and_organisation. A person_and_organisation defines the initiator for a change_request.

4.3.12 change_request to person_and_organisation

Each change_request has addressee defined by zero or one person_and_organisation. A person_and_organisation defines the addressee for a change_request.

4.3.13 change_state to change

Each change_state has change_reference defined by exactly one change. A change defines the change_reference for a change_state.

4.3.14 change_state to date_and_time

Each change_state has date_time defined by exactly one date_and_time. A date_and_time defines the date_time for a change_state.

4.3.15 change_state to person_and_organisation

Each change_state has author defined by exactly one person_and_organisation. A person_and_organisation define the author for a change_state.

4.3.16 class_and_statutory_designation to class_notation

Each class_and_statutory_designation has the_class defined by exactly one class_notation. A class_notation defines the_class for a class_and_statutory_designation.

4.3.17 class_and_statutory_designation to regulation

Each class_and_statutory_designation has the_statutory defined by exactly one regulation. A regulation defines the_statutory for a class_and_statutory_designation.

4.3.18 class_notation to organization

Each class_notation has class_society defined by exactly one organization. An organization defines the class_society for a class_notation.

4.3.19 deck_moulded_form_design_parameter to moulded_form

Each deck_moulded_form_design_parameter has defined_for defined by one, up to many, moulded_form. A set of moulded_form objects defines the defined_for, for a deck_moulded_form_design_parameter.

4.3.20 definable_object to definition

Each definable_object has definitions defined by zero, up to many, definition. A set of definition objects define the definitions for a definable_object.

4.3.21 definable_object to global_id

Each definable_object has id defined by exactly one global_id. A global_id defines the id for a definable_object.

4.3.22 definition to definable_object

Each definition has defined_for defined by one, up to many, definable_object. A set of definable_object objects defines the defined_for, for a definition.

4.3.23 definition to global_id

Each definition has id defined by exactly one global_id. A global_id defines the id for a definition.

4.3.24 definition_creation to definition

Each definition_creation has base defined by zero, up to many, definition. A set of definition objects define the base for a definition_creation.

4.3.25 definition_creation to definition

Each definition_creation has subject defined by exactly one definition. A definition defines the subject for a definition_creation.

4.3.26 definition_deletion to definition

Each definition_deletion has subject defined by exactly one definition. A definition defines the subject for a definition_deletion.

4.3.27 definition_modification to definition

Each definition_modification has base defined by one, up to many, definition. A set of definition objects defines the base for a definition_modification.

4.3.28 definition_modification to definition

Each definition_modification has subject defined by exactly one definition. A definition defines the subject for a definition_modification.

4.3.29 definition_relationship to definition

Each definition_relationship has definition_1 defined by exactly one definition. A definition defines the definition_1 for a definition_relationship.

4.3.30 definition_relationship to definition

Each definition_relationship has definition_2 defined by exactly one definition. A definition defines the definition_2 for a definition_relationship.

4.3.31 design_definition to representation

Each design_definition has representations defined by zero, up to many, representation. A set of representation objects define the representations for a design_definition.

4.3.32 displacement_operation to moulded_form

Each displacement_operation has displacement_of_moulded_form_to_add_or_subtract defined by exactly one moulded_form. A moulded_form defines the displacement_of_moulded_form_to_add_or_subtract for a displacement_operation.

4.3.33 document to document_usage_constraint

Each document has document_subset defined by one, up to many, document_usage_constraint. A set of document_usage_constraint objects define the document_subset for a document.

4.3.34 document_usage_constraint to document

Each document_usage_constraint has source defined by exactly one document. A document defines the source for a document_usage_constraint.

4.3.35 envisaged_definition_creation to definition

Each envisaged_definition_creation has base defined by zero, up to many, definition. A set of definition objects defines the base for an envisaged_definition_creation.

4.3.36 event to date_and_time

Each event has caused_when defined by exactly one date_and_time. A date_and_time defines the caused_when for an event.

4.3.37 event to person_and_organisation

Each event has caused_by defined by exactly one person_and_organisation. A person_and_organisation define the caused_by for an event.

4.3.38 extended_conversion_based_unit to extended_measure_with_unit

Each extended_conversion_based_unit has conversion_factor defined by exactly one extended_measure_with_unit. An extended_measure_with_unit defines the conversion_factor for an extended_conversion_based_unit.

4.3.39 external_instance_reference to global_id

Each external_instance_reference has target_guid defined by exactly one global_id. A global_id defines the target_guid for an external_instance_reference.

4.3.40 general_characteristics_definition to ship

Each general_characteristics_definition has defined_for defined by one, up to many ship's. A set of ship objects define the defined_for, for a general_characteristics_definition.

4.3.41 hull_moulded_form_design_parameter to midship_tumble

Each hull_moulded_form_design_parameter has midship_tumble_data defined by zero, or one midship_tumble. A midship_tumble defines the midship_tumble_data for a hull_moulded_form_design_parameter.

4.3.42 hydrostatic_definition to displacement_operation

Each hydrostatic_definition has displacement_changes defined by zero, up to many displacement_operation. A list of displacement_operation objects defines the displacement_changes for a hydrostatic_definition.

4.3.43 hydrostatic_definition to hydrostatic_table

Each hydrostatic_definition has representations defined by one, up to many, hydrostatic_table. A set of hydrostatic_table objects define the representations for a hydrostatic_definition.

4.3.44 hydrostatic_definition to ship_moulded_form

Each hydrostatic_definition has defined_for defined by one, up to many, ship_moulded_form. A set of ship_moulded_form objects define the defined_for for a hydrostatic_definition.

4.3.45 hydrostatic_position_value to cartesian_point

Each hydrostatic_position_value has position defined by exactly one cartesian_point. A cartesian_point defines the position for a hydrostatic_position_value.

4.3.46 hydrostatic_properties_for_constant_floating_position to floating_position

Each hydrostatic_properties_for_constant_floating_position has definition_of_floating_position defined by exactly one floating_position. A floating_position defines the definition_of_floating_position for a hydrostatic_properties_for_constant_floating_position.

4.3.47 hydrostatic_properties_for_constant_floating_position to hydrostatic_property_value

Each hydrostatic_properties_for_constant_floating_position has hydrostatic_property_values defined by one, up to many, hydrostatic_property_value. A list of hydrostatic_property_value objects define the hydrostatic_property_values for a hydrostatic_properties_for_constant_floating_position.

4.3.48 hydrostatic_properties_for_constant_floating_position to hydrostatic_table

Each hydrostatic_properties_for_constant_floating_position has related_hydrostatic_table defined by one, up to many, hydrostatic_table. A set of hydrostatic_table objects define the related_hydrostatic_table for a hydrostatic_properties_for_constant_floating_position.

4.3.49 hydrostatic_property to extended_measure_with_unit

Each hydrostatic_property has property_unit defined by exactly one extended_measure_with_unit. An extended_measure_with_unit defines the property_unit for a hydrostatic_property.

4.3.50 hydrostatic_table to hydrostatic_properties_for_constant_floating_position

Each hydrostatic_table has items defined by one, up to many, hydrostatic_properties_for_constant_floating_position. A set of hydrostatic_properties_for_constant_floating_position objects define the items for a hydrostatic_table.

4.3.51 hydrostatic_table to hydrostatic_property

Each hydrostatic_table has properties_in_the_hydrostatic_table defined by one, up to many, hydrostatic_property. A list of hydrostatic_property objects define the properties_in_the_hydrostatic_table for a hydrostatic_table.

4.3.52 item to external_reference

Each item has documentation defined by zero, up to many, external_reference. A set of external_reference objects define the documentation for an item.

4.3.53 item to ship

Each item has ship_context defined by zero or one ship. A ship defines the ship_context for an item.

4.3.54 item_relationship to external_instance_reference

Each item_relationship has external_item_1 defined by zero or one external_instance_reference. An external_instance_reference defines the external_item_1 for an item_relationship.

4.3.55 item_relationship to external_instance_reference

Each item_relationship has external_item_2 defined by zero or one external_instance_reference. An external_instance_reference defines the external_item_2 for an item_relationship.

4.3.56 item_relationship to item

Each item_relationship has item_1 defined by zero or one item. An item defines the item_1 for an item_relationship.

4.3.57 item_relationship to item

Each item_relationship has item_2 defined by zero or one item. An item defines the item_2 for an item_relationship.

4.3.58 item_structure to external_instance_reference

Each item_structure has external_items defined by zero, up to many, external_instance_reference. A set of external_instance_reference objects define the external_items for an item_structure.

4.3.59 item_structure to external_instance_reference

Each item_structure has external_relationships defined by zero, up to many, external_instance_reference. A set of external_instance_reference objects define the external_relationships for an item_structure.

4.3.60 item_structure to item

Each item_structure has items defined by zero, up to many items. A set of item objects defines the items for an item_structure.

4.3.61 item_structure to item_relationship

Each item_structure has relationships defined by zero, up to many, item_relationship. A set of item_relationship objects define the relationships for an item_structure.

4.3.62 knot to ship_curve

Each knot has intersecting_ship_curves defined by two, up to many, ship_curve. A list of ship_curve objects define the intersecting_ship_curves for a knot.

4.3.63 knot to vector

Each knot has tangent_information defined by zero or one vector. A vector defines the tangent_information for a knot.

4.3.64 local_co_ordinate_system_with_position_reference to cartesian_point

Each local_co_ordinate_system_with_position_reference has location defined by exactly one cartesian_point. A cartesian_point defines the location for a local_co_ordinate_system_with_position_reference.

4.3.65 longitudinal_table to longitudinal_position

Each longitudinal_table has spacing_table_representations defined by zero, up to many, longitudinal_position. A list of longitudinal_position objects define the spacing_table_representations for a longitudinal_table.

4.3.66 moulded_form_characteristics_definition to moulded_form

Each moulded_form_characteristics_definition has defined_for defined by one, up to many, moulded_form. A set of moulded_form objects define the defined_for for a moulded_form_characteristics_definition.

4.3.67 moulded_form_design_definition to moulded_form

Each moulded_form_design_definition has defined_for defined by one, up to many, moulded_form. A set of moulded_form objects define the defined_for for a moulded_form_design_definition.

4.3.68 moulded_form_functional_definition to moulded_form

Each moulded_form_functional_definition has defined_for defined by one, up to many, moulded_form. A set of moulded_form objects define the defined_for for a moulded_form_functional_definition.

4.3.69 moulded_form_relationship to moulded_form

Each moulded_form_relationship has item_1 defined by exactly one moulded_form. A moulded_form defines the item_1 for a moulded_form_relationship.

4.3.70 moulded_form_relationship to moulded_form

Each moulded_form_relationship has item_2 defined by exactly one moulded_form. A moulded_form defines the item_2 for a moulded_form_relationship.

4.3.71 moulded_form_representation_relationship to moulded_form_shape_representation

Each moulded_form_representation_relationship has rep_1 defined by exactly one moulded_form_shape_representation. A moulded_form_shape_representation defines the rep_1 for a moulded_form_representation_relationship.

4.3.72 moulded_form_representation_relationship to moulded_form_shape_representation

Each moulded_form_representation_relationship has rep_2 defined by exactly one moulded_form_shape_representation. A moulded_form_shape_representation defines the rep_2 for a moulded_form_representation_relationship.

4.3.73 moulded_form_shape_representation to symmetry

Each moulded_form_shape_representation has moulded_form_symmetry defined by zero or one symmetry. A symmetry defines the moulded_form_symmetry for a moulded_form_shape_representation.

4.3.74 offset_point_table_model to section_of_offset_point_table

Each offset_point_table_model has offset_point_table_sections defined by one, up to many, section_of_offset_point_table. A list of section_of_offset_point_table objects define the offset_point_table_sections for an offset_point_table_model.

4.3.75 offset_table_shape_representation to offset_point_table_model

Each offset_table_shape_representation has items defined by one, up to many, offset_point_table_model. A set of offset_point_table_model objects define the items for an offset_table_shape_representation.

4.3.76 owner_designation to organization

Each owner_designation has

defined by exactly one organization. An organization defines the ordering_company for an owner_designation.

4.3.77 owner_designation to organization

Each owner_designation has managing_company defined by exactly one organization. An organization defines the managing_company for an owner_designation.

4.3.78 owner_designation to organization

Each owner_designation has owning_company defined by exactly one organization. An organization defines the owning_company for an owner_designation.

4.3.79 planar_symmetry to plane

Each planar_symmetry has the_symmetry_plane defined by exactly one plane. A plane defines the_symmetry_plane for a planar_symmetry.

4.3.80 propeller_location to cartesian_point

Each propeller_location has shaft_line_location defined by exactly one cartesian_point. A cartesian_point defines the shaft_line_location for a propeller_location.

4.3.81 propeller_location to cartesian_point

Each propeller_location has propeller_location defined by exactly one cartesian_point. A cartesian_point defines the propeller_location for a propeller_location.

4.3.82 propeller_moulded_form_design_parameter to propeller_location

Each propeller_moulded_form_design_parameter has location_of_the_propeller_at_the_ship_hull defined by zero or one propeller_location. A propeller_location defines the location_of_the_propeller_at_the_ship_hull for a propeller_moulded_form_design_parameter.

4.3.83 Regulation to document_reference

Each regulation has international_regulations defined by zero, up to many, document_reference. A set of document_reference objects define the international_regulations for a regulation.

4.3.84 regulation to document_reference

Each regulation has national_regulations defined by zero, up to many, document_reference. A set of document_reference objects define the national_regulations for a regulation.

4.3.85 regulation to document_reference

Each regulation has standards defined by zero, up to many, document_reference. A set of document_reference objects define the standards for a regulation.

4.3.86 revision to versionable_object

Each revision has members defined by one, up to many, versionable_object. A set of versionable_object objects define the members for a revision.

4.3.87 revision_with_context to definable_object

Each revision_with_context has context_of_revision defined by exactly one definable_object. A definable_object defines the context_of_revision for a revision_with_context.

4.3.88 rotational_symmetry to axis1_placement

Each rotational_symmetry has the_rotational_axis defined by exactly one axis1_placement. An axis1_placement defines the_rotational_axis for a rotational_symmetry.

4.3.89 rudder_moulded_form_design_parameter to cartesian_point

Each rudder_moulded_form_design_parameter has rudder_location defined by exactly one cartesian_point. A cartesian_point defines the rudder_location for a rudder_moulded_form_design_parameter.

4.3.90 section_of_offset_point_table to offset_point_table_model

Each section_of_offset_point_table has for_table defined by one, up to many, offset_point_table_model. A set of offset_point_table_model objects define the for_table for a section_of_offset_point_table.

4.3.91 section_of_offset_point_table to ship_point

Each section_of_offset_point_table has section_points defined by one, up to many, ship_point. A list of ship_point objects define the section_points for a section_of_offset_point_table.

4.3.92 ship to item

Each ship has ship_items defined by zero, up to many item. A set of item objects define the ship_items for a ship.

4.3.93 ship_curve to curve

Each ship_curve has curve_shape defined by exactly one curve. A curve defines the curve_shape for a ship_curve.

4.3.94 ship_curve_segment to ship_curve

Each ship_curve_segment has part_of_ship_curve defined by exactly one ship_curve. A ship_curve defines the part_of_ship_curve for a ship_curve_segment.

4.3.95 ship_curve_with_spacing_position to spacing_position

Each ship_curve_with_spacing_position has location defined by exactly one spacing_position. A spacing_position defines the location for a ship_curve_with_spacing_position.

4.3.96 ship_moulded_form to moulded_form

Each ship_moulded_form has items defined by one, up to many, moulded_form. A set of moulded_form objects define the items for a ship_moulded_form.

4.3.97 ship_moulded_form to moulded_form_relationship

Each ship_moulded_form has relationships defined by zero, up to many, moulded_form_relationship. A set of moulded_form_relationship objects define the relationships for a ship_moulded_form.

4.3.98 ship_moulded_form_revision to moulded_form_design_definition

Each ship_moulded_form_revision has members defined by one, up to many, moulded_form_design_definition. A set of moulded_form_design_definition objects define the members for a ship_moulded_form_revision.

4.3.99 ship_moulded_form_revision to ship_moulded_form

Each ship_moulded_form_revision has context_of_revision defined by exactly one ship_moulded_form. A ship_moulded_form defines the context_of_revision for a ship_moulded_form_revision.

4.3.100 ship_moulded_form_wireframe_model to ship_curve

Each ship_moulded_form_wireframe_model has mesh_definition_ship_curves defined by four, up to many, ship_curve. A list of ship_curve objects define the mesh_definition_ship_curves for a ship_moulded_form_wireframe_model.

4.3.101 ship_overall_dimensions to ship_moulded_form

Each ship_overall_dimensions has defined_for defined by one, up to many, ship_moulded_form. A set of ship_moulded_form objects define the defined_for for a ship_overall_dimensions.

4.3.102 ship_point to point

Each ship_point has point_shape defined by exactly one point. A point defines the point_shape for a ship_point.

4.3.103 ship_surface to surface

Each ship_surface has surface_shape defined by exactly one surface. A surface defines the surface_shape for a ship_surface.

4.3.104 shiptype to ship

Each shiptype has defined_for defined by one, up to many ship. A set of ship objects define the defined_for for a shiptype.

4.3.105 shipyard_designation to organization

Each shipyard_designation has shipyard defined by exactly one organization. An organization defines the shipyard for a shipyard_designation.

4.3.106 spacing_position_with_offset to spacing_position

Each spacing_position_with_offset has relating_spacing_position defined by exactly one spacing_position. A spacing_position defines the relating_spacing_position for a spacing_position_with_offset.

4.3.107 spacing_table to spacing_position

Each spacing_table has spacing_table_representations defined by zero, up to many, spacing_position. A list of spacing_position objects define the spacing_table_representations for a spacing_table.

4.3.108 stability_definition to ship_moulded_form

Each stability_definition has defined_for defined by one, up to many, ship_moulded_form. A set of ship_moulded_form objects define the defined_for for a stability_definition.

4.3.109 stability_definition to stability_table

Each stability_definition has representations defined by one, up to many, stability_table. A set of stability_table objects define the representations for a stability_definition.

4.3.110 stability_properties_for_one_floating_position to cartesian_point

Each stability_properties_for_one_floating_position has centre_of_gravity_above_keel defined by exactly one cartesian_point. A cartesian_point defines the centre_of_gravity_above_keel for a stability_properties_for_one_floating_position.

4.3.111 stability_properties_for_one_floating_position to floating_position

Each stability_properties_for_one_floating_position has definition_of_starting_floating_position defined by exactly one floating_position. A floating_position defines the definition_of_starting_floating_position for a stability_properties_for_one_floating_position.

4.3.112 stability_properties_for_one_floating_position to stability_property

Each stability_properties_for_one_floating_position has stability_properties_for_different_angles_of_heel defined by one, up to many, stability_property. A list of stability_property objects define the stability_properties_for_different_angles_of_heel for a stability_properties_for_one_floating_position.

4.3.113 stability_properties_for_one_floating_position to stability_table

Each stability_properties_for_one_floating_position has related_stability_table defined by one, up to many, stability_table. A set of stability_table objects define the related_stability_table for a stability_properties_for_one_floating_position.

4.3.114 stability_property to cartesian_point

Each stability_property has centre_of_buoyancy defined by exactly one cartesian_point. A cartesian_point defines the centre_of_buoyancy for a stability_property.

4.3.115 stability_table to stability_properties_for_one_floating_position

Each stability_table has items defined by one, up to many, stability_properties_for_one_floating_position. A set of stability_properties_for_one_floating_position objects define the items for a stability_table.

4.3.116 surface_shape_representation to face_based_surface_model

Each surface_shape_representation has items defined by one, up to many, face_based_surface_model. A set of face_based_surface_model objects define the items for a surface_shape_representation.

4.3.117 thruster_moulded_form_design_parameter to cartesian_point

Each thruster_moulded_form_design_parameter has geometric_thruster_location defined by exactly one cartesian_point. A cartesian_point defines the geometric_thruster_location for a thruster_moulded_form_design_parameter.

4.3.118 thruster_moulded_form_design_parameter to propeller_moulded_form_design_parameter

Each thruster_moulded_form_design_parameter has thruster_propeller_parameter defined by exactly one propeller_moulded_form_design_parameter. A propeller_moulded_form_design_parameter defines the thruster_propeller_parameter for a thruster_moulded_form_design_parameter.

4.3.119 transversal_table to transversal_position

Each transversal_table has spacing_table_representations defined by zero, up to many, transversal_position. A list of transversal_position objects define the spacing_table_representations for a transversal_table.

4.3.120 version_history to version_relationship

Each version_history has relationships defined by zero, up to many, version_relationship. A set of version_relationship objects define the relationships for a version_history.

4.3.121 version_history to versionable_object

Each version_history has versions defined by one, up to many, versionable_object. A set of versionable_object objects define the versions for a version_history.

4.3.122 version_history to versionable_object

Each version_history has current_version defined by exactly one versionable_object. A versionable_object defines the current_version for a version_history.

4.3.123 version_relationship to versionable_object

Each version_relationship has predecessor defined by exactly one versionable_object. A versionable_object defines the predecessor for a version_relationship.

4.3.124 version_relationship to versionable_object

Each version_relationship has successor defined by exactly one versionable_object. A versionable_object defines the successor for a version_relationship.

4.3.125 vertical_table to vertical_position

Each vertical_table has spacing_table_representations defined by zero, up to many, vertical_position. A list of vertical_position objects define the spacing_table_representations for a vertical_table.

4.3.126 wireframe_shape_representation to shell_based_wireframe_model

Each wireframe_shape_representation has items defined by one, up to many, shell_based_wireframe_model. A set of shell_based_wireframe_model objects define the items for a wireframe_shape_representation.

5. Application interpreted model

5.1 Mapping table

This clause contains the mapping table that shows how each UoF and application object of this part of ISO 10303 (see clause 4) maps to one or more AIM constructs (see annex A). The mapping table is organized in five columns.

Column 1) Application element: Name of an application element as it appears in the application object definition in 4.2. Application object names are written in uppercase. Attribute names and assertions are listed after the application object to which they belong and are written in lower case.

Column 2) AIM element: Name of an AIM element as it appears in the AIM (see annex A), the term 'IDENTICAL MAPPING', or the term 'PATH'. AIM entities are written in lower case. Attribute names of AIM entities are referred to as <entity name>.<attribute name>. The mapping of an application element may result in several related AIM elements. Each of these AIM elements requires a line of its own in the table. The term 'IDENTICAL MAPPING' indicates that both application objects of an application assertion map to the same AIM element. The term 'PATH' indicates that the application assertion maps to the entire reference path.

Column 3) Source: For those AIM elements that are interpreted from the integrated resources or the application interpreted constructs, this is the number of the corresponding part of ISO 10303. For those AIM elements that are created for the purpose of this part of ISO 10303, this is the number of this part. Entities or types that are defined within the integrated resources have an AIC as the source reference if the use of the entity or type for the mapping is within the scope of AIC..

Column 4) Rules: One or more numbers may be given that refer to rules that apply to the current AIM element or reference path. For rules that are derived from relationships between application objects, the same rule is referred to by the mapping entries of all the involved AIM elements. The expanded names of the rules are listed after the table.

Column 5) Reference path: To describe fully the mapping of an application object, it may be necessary to specify a reference path through several related AIM elements. The reference path column documents the role of an AIM element relative to the AIM element in the row succeeding it. Two or more such related AIM elements define the interpretation of the integrated resources that satisfies the requirement specified by the application object. For each AIM element that has been created for use within this part of ISO 10303, a reference path up to its supertype from an integrated resource is specified.

For the expression of reference paths and the relationships between AIM elements the following notational conventions apply:

[]: enclosed section constrains multiple AIM elements or sections of the reference path are required to satisfy an information requirement;

(): enclosed section constrains multiple AIM elements or sections of the reference path are identified as alternatives within the mapping to satisfy an information requirement;

{}: enclosed section constrains the reference path to satisfy an information requirement;

<>: enclosed section constrains at one or more required reference path;

''': enclosed section constrains the supertype entity;

->: attribute references the entity or select type given in the following row;

<-: entity or select type is referenced by the attribute in the following row;

[i]: attribute is an aggregation of which a single member is given in the following row;

[n]: attribute is an aggregation of which member n is given in the following row;

=>: entity is a supertype of the entity given in the following row;

<=: entity is a subtype of the entity given in the following row;

=: the string, select, or enumeration type is constrained to a choice or value;

\: the reference path expression continues on the next line.

Figure 53 on the following page shows the key mappings for this part of 10303 Ship Moulded Forms.

NOTE - The mapping tables of this part of ISO 10303 are making use of some mapping table simplifications, which are explained in detail in the AP 216 Interpretation Report. First templates are used for mapping paths, which are used frequently. Further attributes are mapped either at the super- or at the subtype.

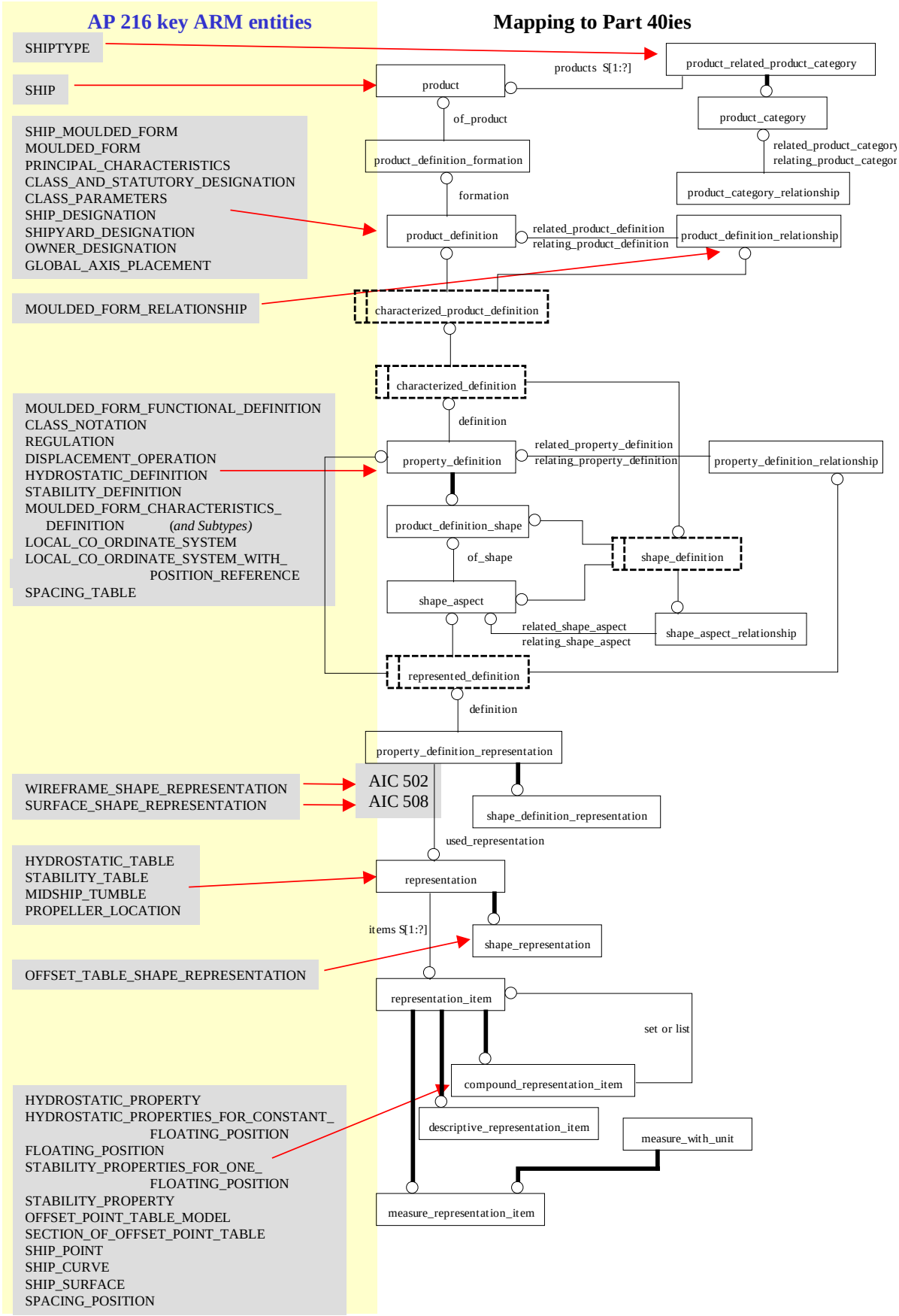


Figure 53 – Key mappings for AP 216

Table 1 - Mapping table for ship_moulded_forms UoF

Application Element	AIM Element	Source	Rules	Reference Path
SHIP_MOULDED_FORM	[product_definition] [group]	41 41		{[CLASS<product_definition, ‘ship moulded form’, ‘item structure‘>] [CLASS<product_definition, ‘item structure‘, ‘definable object’>] [CLASS<product_definition, ‘item structure‘, ‘versionable object’>] [CLASS<product_definition, ‘ship moulded form’, ‘item‘>] [CLASS<product_definition, ‘item‘,‘definable object’>] [ROOT_CLASS<product_definition, ‘definable object’>] [ROOT_CLASS<product_definition, ‘versionable object’>]} LINK_TO_GROUP<product_definition>
ship_moulded_form to moulded_form (as items)	PATH		38	group <- GROUPS<product_definition, ‘item structure‘> {CLASS_ID<product_definition, ‘moulded form’>}
ship_moulded_form to moulded_form_relationship (as relationships)	PATH		39	group <- GROUPS<product_definition_relationship, ‘item structure‘> {CLASS_ID<product_definition_relationship, ‘moulded form relationship’>}
from ITEM				
name		41		*see supertype mappings*
ship_moulded_form to external_reference (as documentation)				*see supertype mappings*
description				*see supertype mappings*
ship_moulded_form to ship (as ship_context)				*see supertype mappings*
from DEFINABLE_OBJECT				
ship_moulded_form to global_id (as id)				*see supertype mappings*
ship_moulded_form to definition (as definitions)				*see supertype mappings*
from ITEM_STRUCTURE				
ship_moulded_form to external_instance_reference (as external_items)				

Application Element	AIM Element	Source	Rules	Reference Path
ship_moulded_form to external_instance_reference (as external_relationships)				
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
MOULDED_FORM	product_definition	41		{[CLASS<product_definition, ‘moulded form’, ‘item‘>] [CLASS<product_definition, ‘item‘,‘definable object’>] [ROOT_CLASS<product_definition, ‘definable object’>]}
from ITEM				
name				*see supertype mappings*
moulded_form to external_reference (as documentation)				*see supertype mappings*
description				*see supertype mappings*
moulded_form to ship (as ship_context)				*see supertype mappings*
from DEFINABLE_OBJECT				*see supertype mappings*
moulded_form to global_id (as id)				*see supertype mappings*
moulded_form to definition (as definitions)				*see supertype mappings*
MOULDED_FORM_RELATIONSHIP	product_definition_relationship	41		{[CLASS<product_definition_relationship, ‘moulded form relationship’, ‘item relationship‘>] [CLASS<product_definition_relationship, ‘item relationship‘, ‘definable object’>] [CLASS<product_definition_relationship, ‘item relationship‘, ‘versionable object’>] [ROOT_CLASS<product_definition_relationship, ‘definable object’>] [ROOT_CLASS<product_definition_relationship, ‘versionable object’>]}
moulded_form_relationship to moulded_form (as item_1)	PATH		40	product_definition_relationship product_definition_relationship.related_product_definition -> product_definition {CLASS_ID<product_definition, ‘moulded form’>}

Application Element	AIM Element	Source	Rules	Reference Path
moulded_form_relationship to moulded_form (as item_2)	PATH		41	product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition {CLASS_ID<product_definition, ‘moulded form’>}
from ITEM_RELATIONSHIP				
ship_moulded_form_relationship to external_instance_reference (as external_item1)				
ship_moulded_form_relationship to external_instance_reference (as external_item2)				
contxt				*see supertype mappings*
from DEFINABLE_OBJECT				
moulded_form_relationship to global_id (as id)				*see supertype mappings*
moulded_form_relationship to definition (as definitions)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
SHIP_MOULDED_FORM_REVISION	group	41		{[CLASS<group, ‘ship moulded form revision’, ‘revision with context‘>] [CLASS<group, ‘revision with context ‘, ‘revision’>] [ROOT_CLASS<group, ‘revision’>]}
ship_moulded_form_revision to ship_moulded_form (as context_of_revision)	PATH			group <- GROUPS<product_definition, ‘context of revision‘> {CLASS_ID<product_definition, ‘ship moulded form’>}
ship_moulded_form_revision to moulded_form_design_definition (as members)	PATH			group <- GROUPS<product_definition_shape, ‘members of revision‘> {CLASS_ID<product_definition_shape, ‘moulded form design definition’>}
from REVISION				
name	group.name	41		
reason	group.description	41		
MOULDED_FORM_FUNCTIONAL_DEFI	extended_property_definition	216	112	{[CLASS<extended_property_definition, ‘moulded form functional

Application Element	AIM Element	Source	Rules	Reference Path
NITION				definition’, ‘functional definition’>] [CLASS<extended_property_definition, ‘functional definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
the_function	descriptive_representation_item .description	45	42, 81	extended_property_definition <= PROP_DEF_REP_HELP<‘moulded form function parameters’> representation representation.items [i] -> representation_item => {representation_iterm.name = ‘function‘} descriptive_representation_item {(descriptive_representation_item.description = ‘ship hull‘) (descriptive_representation_item.description = ‘propeller‘) (descriptive_representation_item.description = ‘rudder‘) (descriptive_representation_item.description = ‘bulbous bow‘) (descriptive_representation_item.description = ‘bulbous stern‘) (descriptive_representation_item.description = ‘appendage‘) (descriptive_representation_item.description = ‘thruster‘) (descriptive_representation_item.description = ‘keel‘) (descriptive_representation_item.description = ‘transom‘) (descriptive_representation_item.description = ‘double bottom‘) (descriptive_representation_item.description = ‘double ship hull‘) (descriptive_representation_item.description = ‘deck‘) (descriptive_representation_item.description = ‘superstructure‘) (descriptive_representation_item.description = ‘horizontal girder‘) (descriptive_representation_item.description = ‘longitudinal girder‘) (descriptive_representation_item.description = ‘longitudinal bulkhead‘) (descriptive_representation_item.description = ‘transverse bulkhead‘) (descriptive_representation_item.description = ‘frame‘) (descriptive_representation_item.description = ‘pressure hull‘)

Application Element	AIM Element	Source	Rules	Reference Path
				(descriptive_representation_item.description = ‘user defined‘)}
moulded_form_functional_definition to moulded_form (as defined_for)	PATH			extended_property_definition <= PROP_TO_PROD_DEF {CLASS_ID<product_definition, ‘ship moulded form’>}
from FUNCTIONAL_DEFINITION				
user_def_function	descriptive_representation_item .description	41	86	extended_property_definition <= PROP_DEF_REP_HELP<‘moulded form function parameters’> representation representation.items [i] -> representation_item => {representation_iterm.name = ‘user def function‘} descriptive_representation_item descriptive_representation_item.description
moulded_form_functional_definition to unit (as local_units)				
from DEFINITION				
moulded_form_functional_definition to global_id (as id)	extended_property_definition.id	216	63, 43	
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
MOULDED_FORM_DESIGN_DEFINITION	extended_product_definition_shape	216		{[CLASS<extended_product_definition_shape, ‘moulded form design definition’, ‘design definition’>] [CLASS<extended_product_definition_shape, ‘design definition’, ‘definition’>] [CLASS<extended_product_definition_shape, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_product_definition_shape, ‘versionable object’>]}
moulded_form_design_definition to moulded_form (as defined_for)	PATH			extended_product_definition_shape <= product_definition_shape <= PROP_TO_PROD_DEF

Application Element	AIM Element	Source	Rules	Reference Path
				{CLASS_ID<product_definition, ‘moulded form’>}
from DESIGN_DEFINITION				
moulded_form_design_definition to representation (as representations)				extended_product_definition_shape <= product_definition_shape <- shape_aspect.of_shape shape_aspect SA2REP_D<‘moulded form design representation‘> representation
from DEFINITION				
moulded_form_design_definition to global_id (as id)	extended_product_definition_shape.id	216	63, 43	
moulded_form_design_definition to unit (as local_units)	PATH			extended_product_definition_shape <= product_definition_shape <- shape_aspect.of_shape shape_aspect SA2REP_D<‘moulded form design representation‘> representation representation.context_of_items -> representation_context => global_unit_assigned_context global_unit_assigned_context.units
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_product_definition_shape>
MOULDED_FORM_SHAPE_REPRESENTATION	shape_representation	41		{CLASS_ID<shape_representation, ‘moulded form shape representation’>}
moulded_form_representation_id	representation.id	43		

Application Element	AIM Element	Source	Rules	Reference Path
moulded_form_shape_representation to symmetry (as moulded_form_symmetry)	PATH			shape_representation <= representation representation.items[i] -> representation_item => ({representation_item.name = ‘rotational axis’} geometric_representation_item => placement => axis1_placement) ({representation_item.name = ‘symmetry plane’} geometric_representation_item => surface => elementary_surface => plane)
from REPRESENTATION				
name	representation.name	43		
moulded_form_shape_representation to representation_item (as items)	PATH			shape_representation <= representation representation.items[i] -> representation_item
moulded_form_shape_representation to representation_context (as representation_context)	PATH			shape_representation <= representation representation.context_of_items -> representation_context
MOULDED_FORM_REPRESENTATION_RELATIONSHIP	representation_relationship	43		{CLASS_ID<representation_relationship, ‘moulded form representation relationship’>}
id	representation_relationship.name	43		
moulded_form_representation_relationship to moulded_form_representation (as rep_1)	PATH	43		representation_relationship representation_relationship.rep_1 -> representation => shape_representation {CLASS_ID<shape_representation, ‘moulded form shape representation’>}
moulded_form_representation_	PATH	43		representation_relationship

Application Element	AIM Element	Source	Rules	Reference Path
relationship to moulded_form_representation (as rep_1)				representation_relationship.rep_2 -> representation => shape_representation {CLASS_ID<shape_representation, 'moulded form shape representation'>}
from REPRESENTATION_RELATIONSHIP				
name	representation_relationship.name	43		
description	representation_relationship.description	43		
MOULDED_FORM_REPRESENTATION_ITEM	representation_item	43		*see subtype mappings*
from REPRESENTATION_ITEM				
name	representation_item.name	43		
PLANAR_SYMMETRY	plane	42		
planar_symmetry to plane (as the_symmetry_plane)	IDENTICAL MAPPING			
ROTATIONAL_SYMMETRY	axis1_placement	42		
rotational_symmetry to axis1_placement (as the_the_rotational_axis)	IDENTICAL MAPPING			
SYMMETRY	[plane] [axis1_placement]	42		*see subtype mappings*

Table 2 - Mapping table for ship_general_characteristics UoF

Application Element	AIM Element	Source	Rules	Reference Path
PRINCIPAL_CHARACTERISTICS	product_definition	41	4	{[CLASS<product_definition, ‘principal characteristics’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]}
length_between_perpendiculars	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘length between perpendiculars‘, positive_length_measure>
moulded_breadth	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘moulded breadth‘, positive_length_measure>
moulded_depth	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘moulded depth‘, positive_length_measure>
design_draught	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘design draught‘, positive_length_measure>
block_coefficient	value_representation_item.value_component	43	61, 87	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘block coefficient‘, ratio_measure>
design_deadweight	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘design deadweight‘, mass_measure>
min_draught_at_fp	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘min daught at fp‘, positive_length_measure>
max_draught_at_fp	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘max daught at fp‘, positive_length_measure>
min_draught_at_ap	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘min daught at ap‘, positive_length_measure>
max_draught_at_ap	value_representation_item.value_component	43	61, 82	PROD_DEF_TO_VAL_REP_ITEM<‘principal characteristics’, ‘max daught at ap‘, positive_length_measure>
from GENERAL_CHARACTERISTICS_DEFINITION				
principal_characteristics to ship (as defined_for)				*see supertype mappings*

Application Element	AIM Element	Source	Rules	Reference Path
from DEFINITION				
principal_characteristics to global_id (as id)	product_definition.id	41	63, 49	
principal_characteristics to unit (as local_units)	PATH			PROD_DEF_TO_UNITS<‘principal characteristics’>
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>
CLASS_AND_STATUTORY_DESIGNATION	product_definition	41	1	{[CLASS<product_definition, ‘class and statutory designation’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]}
class_number	descriptive_representation_item.description	45	74	PROD_DEF_TO_DESC_REP_ITEM<‘class and statutory designation’, ‘class number‘>
class_and_statutory_designation to class_notation (as the_class)	PATH		64	PROD_DEF_PROP_DEF {CLASS_ID<property_definition, ‘class notation’>}
class_and_statutory_designation to regulation (as the_statutory)	PATH		65	PROD_DEF_PROP_DEF {CLASS_ID<property_definition, ‘regulation’>}
from GENERAL_CHARACTERISTICS_DEFINITION				
class_and_statutory_designation to ship (as defined_for)				*see supertype mappings*
from DEFINITION				
class_and_statutory_designation to global_id (as id)	product_definition.id	41	63, 45	
class_and_statutory_designation to unit (as local_units)				
from VERSIONABLE_OBJECT				

Application Element	AIM Element	Source	Rules	Reference Path
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>
CLASS_NOTATION	property_definition	41	106	{CLASS_ID<property_definition, ‘class notation’>}
class_notation to organisation (as class_society)	PATH		105	ORG_ASSGN<property_definition, ‘class society’>
class_notations_hull	descriptive_representation_item.description	45	5	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘class notations hull’>
class_notations_machinery	descriptive_representation_item.description	45	5	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘class notations machinery’>
ice_class_notation	descriptive_representation_item.description	45	84	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘ice class notation‘>
service_area	descriptive_representation_item.description	45	75	PROP_DEF_TO_DESC_REP_ITEM<‘class society’, ‘service area‘>
service_factor	value_representation_item.value_component	43	61, 84	PROP_DEF_TO_VAL_REP_ITEM<‘class notation’, ‘service factor‘, count_measure>
approval_required_for_oil_cargo	descriptive_representation_item.description	45	75	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘approval required for oil cargo‘> {(descriptive_representation_item.description = ‘TRUE‘) (descriptive_representation_item.description = ‘FALSE‘)}
approval_required_loading_unloading_aground	descriptive_representation_item.description	45	75	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘approval required for loading unloading aground‘> {(descriptive_representation_item.description = ‘TRUE‘) (descriptive_representation_item.description = ‘FALSE‘)}
approval_required_loading_unloading_grabs	descriptive_representation_item.description	45	75	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘approval required for unloading grabs‘> {(descriptive_representation_item.description = ‘TRUE‘) (descriptive_representation_item.description = ‘FALSE‘)}
approval_required_for_heavy_cargo	descriptive_representation_item.description	45	84	PROP_DEF_TO_DESC_REP_ITEM<‘class notation’, ‘approval required for heavy cargo‘> {(descriptive_representation_item.description = ‘HC‘) (descriptive_representation_item.description = ‘HC_E‘) (descriptive_representation_item.description = ‘HC_EA‘)}
REGULATION	property_definition	41		{CLASS_ID<property_definition, ‘regulation’>}
regulation to document_reference	PATH	41		DOC_REF<property_definition, ‘international regulations’>

Application Element	AIM Element	Source	Rules	Reference Path
(as international_regulations)				
regulation to document_reference (as standards)	PATH	41		DOC_REF<property_definition, ‘standards’>
regulation to document_reference (as national_regulations)	PATH	41		DOC_REF<property_definition, ‘national regulations’>
CLASS_PARAMETERS	product_definition	41	2	{[CLASS<product_definition, ‘class parameters’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]}
length_class	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_VAL_REP_ITEM<‘class parameters’, ‘length class‘, positive_length_measure>
length_solas	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_VAL_REP_ITEM<‘class parameters’, ‘length solas‘, positive_length_measure>
scantlings_draught	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_VAL_REP_ITEM<‘class parameters’, ‘scantlings draught‘, positive_length_measure>
block_coefficient_class	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_VAL_REP_ITEM<‘class parameters’, ‘block coefficient class‘, ratio_measure>
design_speed_ahead	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_SPECIAL_VAL_REP_ITEM<‘class parameters’, ‘design speed ahead‘, speed_unit>
design_speed_astern	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_SPECIAL_VAL_REP_ITEM<‘class parameters’, ‘design speed astern‘, speed_unit>
froude_number	value_representation_item.value_component	43	61, 76	PROD_DEF_TO_VAL_REP_ITEM<‘class parameters’, ‘froude number‘, ratio_measure>
from GENERAL_CHARACTERISTICS_DEFINITION				
class_parameters to ship (as defined_for)				*see supertype mappings*
from DEFINITION				
class_parameters to global_id (as id)	product_definition.id	41	63, 49	
class_parameters to unit	PATH			PROD_DEF_TO_UNITS<‘class parameters’>

Application Element	AIM Element	Source	Rules	Reference Path
(as local_units)				
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>
SHIP_DESIGNATION	product_definition	41		{[CLASS<product_definition, ‘ship designation’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]}
ship_name	product_definition.name	41		
ship_type_description	product_definition.description	41		
ship_identification	identification_assignment.assigned_id	41	6	HAS_ID_2_ROLES<product_definition, ‘IMO number’, ‘pennant hull number’>
call_sign	identification_assignment.assigned_id	41	93	HAS_ID_1_ROLE<product_definition, ‘call sign’>
flag_state	identification_assignment.assigned_id	41	94	HAS_ID_1_ROLE<product_definition, ‘flag state’>
port_of_registration	identification_assignment.assigned_id	41	95	HAS_ID_1_ROLE<product_definition, ‘port of registration’>
from GENERAL_CHARACTERISTICS_DEFINITION				
ship_designation to ship (as defined_for)				*see supertype mappings*
from DEFINITION				
ship_designation to global_id (as id)	product_definition.id	41	63, 50	
ship_designation to unit (as local_units)				
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>

Application Element	AIM Element	Source	Rules	Reference Path
SHIPYARD_DESIGNATION	product_definition	41		{[CLASS<product_definition, ‘shipyard designation’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]}
role	organization_assignment.role	41		ORG_ASS_PART<product_definition> {organization_assignment.role -> organization_role {(organization_.role.name = ‘prime design’) (organization_role.name =‘prime build’) (organization_role.name =‘prime repair’) (organization_role.name =‘prime’) (organization_role.name =‘subcontractor’)}
shipyard_new_building_id	identification_assignment.assigned_id	41		HAS_ID_1_ROLE<product_definition, ‘shipyard new building id’>
shipyard_project_name	organizational_project.name	41		ORG_ASSGN<product_definition, ‘shipyard‘> <- organizational_project.responsible_organizations[i] organizational_project {organizational_project.description = ‘shipyard project name‘} organizational_project organizational_project.name
shipyard_designation to organisation (as shipyard_name)	organization_assignment.assigned_organization	41	96	ORG_ASSGN<product_definition, ‘shipyard‘>
from GENERAL_CHARACTERISTICS_DEFINITION				
shipyard_designation to ship (as defined_for)				*see supertype mappings*
from DEFINITION				
shipyard_designation to global_id (as id)	product_definition.id	41	63, 51	
shipyard_designation to unit (as local_units)				

Application Element	AIM Element	Source	Rules	Reference Path
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>
OWNER_DESIGNATION	product_definition	41		{[CLASS<product_definition, ‘owner designation’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]}
owner_designation to organisation (as managing_company)	PATH		90	ORG_ASSGN<product_definition, ‘managing company‘>
owner_designation to organisation (as ordering_company)	PATH		91	ORG_ASSGN<product_definition, ‘ordering company‘>
owner_designation to organisation (as owning_company)	PATH		92	ORG_ASSGN<product_definition, ‘owning company‘>
from GENERAL_CHARACTERISTICS_DEFINITION				
owner_designation to ship (as defined_for)				*see supertype mappings*
from DEFINITION				
owner_designation to global_id (as id)	product_definition.id	41	63, 48	
owner_designation to unit (as local_units)				
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>
SHIPTYPE	extended_product_related_product_category	216		*see subtype mappings*
description	product_category.description	41		extended_product_related_product_category <= product_related_product_category <=

Application Element	AIM Element	Source	Rules	Reference Path
				product_category product_category.description
shiptype to ship (as defined_for)	PATH			extended_product_related_product_category <= product_related_product_category product_related_product_category.products[i] -> product {CLASS_ID<product, ‘ship’>}
from FUNCTIONAL_DEFINITION				
user_def_function	product_category.name	41		product_category_relationship product_category_relationship.category -> product_category product_category_relationship product_category_relationship.sub_category -> product_category product_category.name
shiptype to unit (as local_units)				
from DEFINITION				
shiptype to global_id (as id)	extended_product_related_product_category.id	216	63, 53	
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_product_related_product_category >
CARRIER	extended_product_related_product_category	216		{[PROD_CAT_NAME<‘carrier‘>] [CLASS<extended_product_related_product_category, ‘ship type’, ‘functional definition’>] [CLASS<extended_product_related_product_category, ‘functional definition’, ‘definition’>] [CLASS<extended_product_related_product_category, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_product_related_product_category, ‘versionable object’>]}
has_type	product_category.name	41		PROD_CAT_NAME<‘carrier‘>

Application Element	AIM Element	Source	Rules	Reference Path
				product_category <- product_category_relationship.category product_category_relationship {product_category_relationship.name = ‘carrier types‘} product_category_relationship.sub_category -> product_category {(product_category.name = ‘Container Carrier‘) (product_category.name = ‘Bulk Carrier’) (product_category.name = ‘Ore Carrier’) (product_category.name = ‘Oil Tanker’) (product_category.name = ‘Roro Vessel’) (product_category.name = ‘Ferry’) (product_category.name = ‘Car Ferry’) (product_category.name = ‘Cruise liner’) (product_category.name = ‘Passenger Vessel’) (product_category.name = ‘Cargo ship carrying passengers’) (product_category.name = ‘Product Tanker’) (product_category.name = ‘Gas carrier’) (product_category.name = ‘Liquefied Gas Tanker’) (product_category.name = ‘Chemical Tanker’) (product_category.name = ‘Chemical Tanker Type 1’) (product_category.name = ‘Tanker for Refrigerated Fruit Juice’) (product_category.name = ‘General Cargo Carrier’) (product_category.name = ‘Dry cargo Vessel’) (product_category.name = ‘Refrigerated cargo carrying ship’) (product_category.name = ‘High Speed Craft Passenger’) (product_category.name = ‘High Speed Craft Cargo’) (product_category.name = ‘Hydrofoil’) (product_category.name = ‘Car Carrier’) (product_category.name = ‘Barge’) (product_category.name = ‘Barge for Deck Loading’) (product_category.name = ‘Barge for Oil’) (product_category.name = ‘Barge for Liquefied Gas’) (product_category.name = ‘Barge Pontoon’)

Application Element	AIM Element	Source	Rules	Reference Path
				(product_category.name = ‘user defined’)}
from SHIPTYPE				
description				*see supertype mappings*
carrier to ship (as defined_for)				*see supertype mappings*
from FUNCTIONAL_DEFINITION				
user_def_function				*see supertype mappings*
carrier to unit (as local_units)				*see supertype mappings*
from DEFINITION				
carrier to global_id (as id)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
WORKING_SHIP	extended_product_related_product_category	216		{[PROD_CAT_NAME<‘working ship‘>] [CLASS<extended_product_related_product_category, ‘ship type’, ‘functional definition’>] [CLASS<extended_product_related_product_category, ‘functional definition’, ‘definition’>] [CLASS<extended_product_related_product_category, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_product_related_product_category, ‘versionable object’>]}
has_type	product_category.name	41		PROD_CAT_NAME<‘working ship‘> product_category <- product_category_relationship.category product_category_relationship {product_category_relationship.name = ‘working ship types‘} product_category_relationship.sub_category -> product_category {(product_category.name = ‘Tug’) (product_category.name = ‘Sealer’) (product_category.name = ‘Fire Fighter’) (product_category.name = ‘Drilling Vessel’)

Application Element	AIM Element	Source	Rules	Reference Path
				(product_category.name = ‘Pipe Laying Vessel’) (product_category.name = ‘Crane Vessel’) (product_category.name = ‘Dredger’) (product_category.name = ‘Supply Vessel’) (product_category.name = ‘Ice breaker’) (product_category.name = ‘Fishing Vessel’) (product_category.name = ‘Floating Dock’) (product_category.name = ‘Pilot Boat’) (product_category.name = ‘Floating Hotel’) (product_category.name = ‘Well Stimulation Vessel’) (product_category.name = ‘Pusher’) (product_category.name = ‘Stern Trawler’) (product_category.name = ‘Reefer’) (product_category.name = ‘Offshore Supply Vessel’) (product_category.name = ‘Oil Production Vessel’) (product_category.name = ‘Oil Storage Vessel’) (product_category.name = ‘Oil Production and Storage Vessel’) (product_category.name = ‘Shuttle tanker’) (product_category.name = ‘FPSO’) (product_category.name = ‘FPGO’) (product_category.name = ‘user defined’)}
from SHIPTYPE				
description				*see supertype mappings*
working_ship to ship (as defined_for)	PATH			*see supertype mappings*
from FUNCTIONAL_DEFINITION				
user_def_function				*see supertype mappings*
working_ship to unit (as local_units)				*see supertype mappings*
from DEFINITION				
working_ship to global_id (as id)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*

Application Element	AIM Element	Source	Rules	Reference Path
RESEARCH_SHIP	extended_product_related_product_category	216		{[PROD_CAT_NAME<‘research ship‘>] [CLASS<extended_product_related_product_category, ‘ship type’, ‘functional definition’>] [CLASS<extended_product_related_product_category, ‘functional definition’, ‘definition’>] [CLASS<extended_product_related_product_category, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_product_related_product_category, ‘versionable object’>]}
has_type	product_category.name	41		PROD_CAT_NAME<‘research ship‘> product_category <- product_category_relationship.category product_category_relationship {product_category_relationship.name = ‘research ship types‘} product_category_relationship.sub_category -> product_category {product_category.name = ‘user defined’}
from SHIPTYPE				
description				*see supertype mappings*
research_ship to ship (as defined_for)	PATH			*see supertype mappings*
from FUNCTIONAL_DEFINITION				
user_def_function				*see supertype mappings*
research_ship to unit (as local_units)				*see supertype mappings*
from DEFINITION				
research_ship to global_id (as id)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
NAVY_SHIP	extended_product_related_product_category	216		{[PROD_CAT_NAME<‘navy ship‘>] [CLASS<extended_product_related_product_category, ‘ship type’, ‘functional definition’>] [CLASS<extended_product_related_product_category, ‘functional

Application Element	AIM Element	Source	Rules	Reference Path
				definition’, ‘definition’>] [CLASS<extended_product_related_product_category, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_product_related_product_category, ‘versionable object’>]}
has_type	product_category.name	41		PROD_CAT_NAME<‘navy ship‘> product_category <- product_category_relationship.category product_category_relationship {product_category_relationship.name = 'navy ship types‘} product_category_relationship.category -> product_category product_category_relationship product_category_relationship.sub_category -> product_category {(product_category.name = ‘AIRCRAFT CARRIER’) (product_category.name = ‘CORVETTE’) (product_category.name = ‘CRUISER’) (product_category.name = ‘DESTROYER’) (product_category.name = ‘FLEET AUXILIARY VESSEL’) (product_category.name = ‘FRIGATE’’) (product_category.name = ‘MINE WARFARE SHIP’’) (product_category.name = ‘PATROL FORCE VESSEL’) (product_category.name = ‘SERVICE CRAFT’) (product_category.name = ‘SUBMARINE’) (product_category.name = ‘AUXILIARY OILER’) (product_category.name = ‘LANDING PLATFORM DOCK’) (product_category.name = ‘LANDING PLATFORM HELICOPTER’) (product_category.name = ‘user defined’)}
from SHIPTYPE				
description				*see supertype mappings*
navy_ship to ship (as defined_for)	PATH			*see supertype mappings*

Application Element	AIM Element	Source	Rules	Reference Path
from FUNCTIONAL_DEFINITION				
user_def_function				*see supertype mappings*
navy_ship to unit (as local_units)				*see supertype mappings*
from DEFINITION				
navy_ship to global_id (as id)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*

Table 3 - Mapping table for ship_design_parameter UoF

Application Element	AIM Element	Source	Rules	Reference Path
MOULDED_FORM_CHARACTERISTICS_DEFINITION	extended_property_definition	216	109	*see subtype mappings*
description	property_definition.description	41		
moulded_form_outer_surface	value_representation_item.value_component	43	80, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘moulded form characteristics definition’, ‘moulded form outer surface‘, area_measure>
moulded_form_displacement	value_representation_item.value_component	43	80, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘moulded form characteristics definition’, ‘moulded form displacement‘, volume_measure>
moulded_form_characteristics_definition to moulded_form (as defined_for)	PATH			extended_property_definition <= PROP_TO_PROD_DEF {CLASS_ID<product_definition, ‘moulded_form’>}
from DEFINITION				
moulded_form_characteristics_definition to global_id (as id)	extended_property_definition.id	216	63, 57	
moulded_form_characteristics_definition to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<‘moulded form characteristics definition’>
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
APPENDAGE_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	97	{[CLASS<extended_property_definition, ‘appendage moulded form design parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
appendage_length	value_representation_item.valu	43	66, 61	extended_property_definition <=

Application Element	AIM Element	Source	Rules	Reference Path
	e_component			PROP_DEF_TO_VAL_REP_ITEM<‘appendage moulded form design parameter’, ‘appendage length‘, positive_length_measure>
appendage_breadth	value_representation_item.value_component	43	66, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘appendage moulded form design parameter’, ‘appendage breadth‘, positive_length_measure>
appendage_depth	value_representation_item.value_component	43	66, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘appendage moulded form design parameter’, ‘appendage depth‘, positive_length_measure>
type_of_appendage	descriptive_representation_item.description	45	66	extended_property_definition <= PROP_DEF_TO_DESC_REP_ITEM<‘appendage moulded form design parameter’, ‘type of appendage‘> {(descriptive_representation_item.description = ‘bilge keel‘) (descriptive_representation_item.description = ‘skeg‘) (descriptive_representation_item.description = ‘shaft strut ‘) (descriptive_representation_item.description = ‘shaft bossings‘) (descriptive_representation_item.description = ‘passive fin‘) (descriptive_representation_item.description = ‘spray rail‘)}
from MOULDED_FORM_CHARACTERISTICS_DEFINITION				*see supertype mappings*
description				*see supertype mappings*
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
appendage_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
appendage_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
appendage_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				

Application Element	AIM Element	Source	Rules	Reference Path
version_id				*see supertype mappings*
BOTTOM_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	98	{[CLASS<extended_property_definition, ‘bottom moulded form design parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
bilge_radius	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘bilge radius‘, positive_length_measure>
rise_of_floor	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘rise of floor‘, positive_length_measure>
aft_end_of_flat_of_bottom	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘aft end of flat of bottom‘, positive_length_measure>
front_end_of_flat_of_bottom	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘front end of flat of bottom‘, positive_length_measure>
flat_of_bottom_breadth	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘flat of bottom breadth‘, positive_length_measure>
length_of_flat_of_bottom	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘length of flat of bottom breadth‘, positive_length_measure>
rake_of_keel	value_representation_item.value_component	43	67, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bottom moulded form design parameter’, ‘rake of keel‘, plane_angle_measure>
from MOULDED_FORM_CHARACTERISTICS_DEFINITION				

Application Element	AIM Element	Source	Rules	Reference Path
description				*see supertype mappings*
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
bottom_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
bottom_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
bottom_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
BULB_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	99, 99	{[CLASS<extended_property_definition, ‘bulb moulded form design parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
bulb_length	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bulb moulded form design parameter’, ‘bulb length‘, positive_length_measure>
bulb_length_from_pp	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bulb moulded form design parameter’, ‘bulb length from pp‘, positive_length_measure>
bulb_breadth	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bulb moulded form design parameter’, ‘bulb breadth‘, positive_length_measure>
bulb_breadth_pp	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘bulb moulded form design

Application Element	AIM Element	Source	Rules	Reference Path
				parameter', 'bulb breadth pp', positive_length_measure>
bulb_depth	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<'bulb moulded form design parameter', 'bulb depth', positive_length_measure>
bulb_depth_pp	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<'bulb moulded form design parameter', 'bulb depth pp', positive_length_measure>
bulb_frame_section_area_at_pp	value_representation_item.value_component	43	68, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<'bulb moulded form design parameter', 'bulb frame section area at pp', area_measure>
bulb_location	descriptive_representation_item.description	45	68, 61	extended_property_definition <= PROP_DEF_TO_DESC_REP_ITEM<'bulb moulded form design parameter', 'bulb location'> {(descriptive_representation_item.description = 'bow') (descriptive_representation_item.description = 'stern')}
from MOULDED_FORM_CHARACTERISTICS_DEFINITION				
description				*see supertype mappings*
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
bulb_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
bulb_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
bulb_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
HULL_MOULDED_FORM_DESIGN_PA	extended_property_definition	216	101	{[CLASS<extended_property_definition, 'hull moulded form design

Application Element	AIM Element	Source	Rules	Reference Path
RAMETER				parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
aft_end_of_parallel_midbody_at_design_draught	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘aft end of parallel midbody at design draught‘, positive_length_measure>
front_end_of_parallel_midbody_at_design_draught	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘front end of parallel midbody at design draught‘, positive_length_measure>
length_of_parallel_midbody_at_design_draught	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘length of parallel midbody at design draught‘, positive_length_measure>
aft_end_of_flat_of_side	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘aft end of flat of side‘, positive_length_measure>
front_end_of_flat_of_side	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘front end of flat of side‘, positive_length_measure>
length_of_flat_of_side	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘length of flat of side‘, positive_length_measure>
block_coefficient	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘block coefficient‘, ratio_measure>
prismatic_coefficient	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘prismatic coefficient‘, ratio_measure>

Application Element	AIM Element	Source	Rules	Reference Path
waterline_angle_of_entrance_at_stern	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘waterline angle of entrance at stern‘, plane_angle_measure>
waterline_angle_of_entrance_at_bow	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘waterline angle of entrance at bow‘, plane_angle_measure>
max_frame_section_area_location	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘max frame section area location‘, positive_length_measure>
max_wetted_frame_section_area	value_representation_item.value_component	43	70, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘max wetted frame section area‘, area_measure>
hull_length_pp	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘hull length pp‘, positive_length_measure>
hull_breadth	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘hull breadth‘, positive_length_measure>
hull_depth	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘hull depth‘, positive_length_measure>
hull_design_draught	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘hull design draught‘, positive_length_measure>
gunwale_radius	value_representation_item.value_component	43	85, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘hull moulded form design parameter’, ‘gunwale radius‘, positive_length_measure>
hull_moulded_form_design_parameter to midship_tumble (as midship_tumble_data)	PATH			extended_property_definition <= PROP_DEF_TO_REP {CLASS_ID<representation, ‘midship_tumble’>}
from				

Application Element	AIM Element	Source	Rules	Reference Path
MOULDED_FORM_CHARACTERISTICS _DEFINITION				
description				*see supertype mappings*
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
hull_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
hull_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
hull_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
MIDSHIP_TUMBLE	representation	43		{CLASS_ID<representation, ‘midship tumble’>}
tumble_out_at_bottom	value_representation_item.value_component	43	22, 61	REP_TO_VAL_REP_ITEM<‘tumble out at bottom‘, positive_length_measure>
tumble_in_at_top	value_representation_item.value_component	43	22, 61	REP_TO_VAL_REP_ITEM<‘tumble in at top‘, positive_length_measure>
tumble_out_at_side	value_representation_item.value_component	43	22, 61	REP_TO_VAL_REP_ITEM<‘tumble out at side‘, positive_length_measure>
tumble_in_at_side	value_representation_item.value_component	43	22, 61	REP_TO_VAL_REP_ITEM<‘tumble in at side‘, positive_length_measure>
SHIP_OVERALL_DIMENSIONS	extended_property_definition	216	110	{[CLASS<extended_property_definition, ‘ship overall dimensions’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
overall_breadth	value_representation_item.valu	43	83, 61	extended_property_definition <=

Application Element	AIM Element	Source	Rules	Reference Path
	e_component			PROP_DEF_TO_VAL_REP_ITEM<‘ship overall dimensions’, ‘overall breadth‘, positive_length_measure>
overall_depth	value_representation_item.value_component	43	83, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘ship overall dimensions’, ‘overall depth‘, positive_length_measure>
overall_length	value_representation_item.value_component	43	83, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘ship overall dimensions’, ‘overall length‘, positive_length_measure>
stem_overhang	value_representation_item.value_component	43	83, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘ship overall dimensions’, ‘stem overhang‘, positive_length_measure>
stern_overhang	value_representation_item.value_component	43	83, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘ship overall dimensions’, ‘stern overhang‘, positive_length_measure>
ship_overall_dimensions to ship_moulded_form (as defined_for)	PATH			extended_property_definition <= PROP_TO_PROD_DEF {CLASS_ID<product_definition, ‘ship moulded_form’>}
from DEFINITION				
ship_overall_dimensions to global_id (as id)	extended_property_definition.id	216	63, 58	
ship_overall_dimensions to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<‘ ship overall dimensions’>
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
PROPELLER_LOCATION	representation	43		{CLASS_ID<representation, ‘propeller location’>}
shaft_line_inclination_x	value_representation_item.value_component	43	23, 61	REP_TO_VAL_REP_ITEM<‘shaft line inclination x‘>
shaft_line_inclination_y	value_representation_item.value_component	43	23, 61	REP_TO_VAL_REP_ITEM<‘shaft line inclination y‘>
propeller_location to cartesian_point (as shaft_line_location)	PATH		23, 61	REP_TO_CART_POINT<‘shaft line location‘>
propeller_location to cartesian_point (as propeller_location)	PATH		23,61	REP_TO_CART_POINT<‘propeller location‘>

Application Element	AIM Element	Source	Rules	Reference Path
PROPELLER_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	102	{[CLASS<extended_property_definition, ‘propeller moulded form design parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
type_of_propulsion	descriptive_representation_item.description	45	71	extended_property_definition <= PROP_DEF_TO_DESC_REP_ITEM<‘propeller moulded form design parameter’, ‘type of propulsion‘> {(descriptive_representation_item.description = ‘water jet‘) (descriptive_representation_item.description = ‘paddle wheel‘) (descriptive_representation_item.description = ‘vertical axis propeller‘) (descriptive_representation_item.description = ‘screw propeller‘) (descriptive_representation_item.description = ‘ducted propeller‘) (descriptive_representation_item.description = ‘thruster‘) (descriptive_representation_item.description = ‘user defined‘)}
propeller_diameter	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘propeller diameter‘, positive_length_measure>
chord_length_at_0_7_radius	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘chord length at 0 7 radius‘, positive_length_measure>
thickness_at_0_7_radius	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘thickness at 0 7 radius‘, positive_length_measure>
number_of_propeller_blades	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘number of propeller blades‘, count_measure>
blade_mean_height	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘blade mean height‘, positive_length_measure>

Application Element	AIM Element	Source	Rules	Reference Path
expanded_area_ratio	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘expanded area ratio‘, ratio_measure>
hub_diameter_ratio	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘hub diameter ratio‘, ratio_measure>
nominal_design_pitch_ratio	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘nominal design pitch ratio‘, ratio_measure>
type_of_propeller_blades	descriptive_representation_item.description	45	71	extended_property_definition <= PROP_DEF_TO_DESC_REP_ITEM<‘propeller moulded form design parameter’, ‘type of propeller blades‘> {(descriptive_representation_item.description = ‘fixed blade‘) (descriptive_representation_item.description = ‘adjustable blade‘) (descriptive_representation_item.description = ‘controllable blade‘)}
rake	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘rake‘, plane_angle_measure>
skew	value_representation_item.value_component	43	71, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘propeller moulded form design parameter’, ‘skew‘, plane_angle_measure>
design_sense_of_rotation	descriptive_representation_item.description	45	71	extended_property_definition <= PROP_DEF_TO_DESC_REP_ITEM<‘propeller moulded form design parameter’, ‘design sense of rotation‘> {(descriptive_representation_item.description = ‘right‘) (descriptive_representation_item.description = ‘left‘)}
propeller_moulded_form_design_parameter to propeller_location (as location_of_the_propeller_at_the_ship_hull)	PATH			extended_property_definition <= PROP_DEF_TO_REP {CLASS_ID<representation, ‘propeller_location’>}
from MOULDED_FORM_CHARACTERISTICS_DEFINITION				
description				*see supertype mappings*

Application Element	AIM Element	Source	Rules	Reference Path
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
propeller_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
propeller_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
propeller_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
RUDDER_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	103	{[CLASS<extended_property_definition, ‘rudder moulded form design parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
rudder_height	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘rudder height‘, positive_length_measure>
rudder_mean_height	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘rudder mean height‘, positive_length_measure>
rudder_length	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘rudder length‘, positive_length_measure>
rudder_mean_length	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘rudder mean length‘, positive_length_measure>

Application Element	AIM Element	Source	Rules	Reference Path
rudder_thickness	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘rudder thickness‘, positive_length_measure>
projected_rudder_area	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘projected rudder area‘, positive_length_measure>
type_of_the_rudder	descriptive_representation_item.description	45	72	extended_property_definition <= PROP_DEF_TO_DESC_REP_ITEM<‘rudder moulded form design parameter’, ‘type of the rudder‘> {(descriptive_representation_item.description = ‘balanced‘) (descriptive_representation_item.description = ‘unbalanced‘)}
rudder_moulded_form_design_parameter to cartesian_point (as rudder_location)	PATH		72, 61	extended_property_definition <= PROP_DEF_TO_CART_POINT<‘rudder moulded form design parameter’, ‘rudder location‘>
aspect_ratio	value_representation_item.value_component	43	72, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘rudder moulded form design parameter’, ‘aspect ratio‘, ratio_measure>
from MOULDED_FORM_CHARACTERISTICS_DEFINITION				
description				*see supertype mappings*
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
rudder_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
rudder_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
rudder_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				

Application Element	AIM Element	Source	Rules	Reference Path
version_id				*see supertype mappings*
THRUSTER_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	104	{[CLASS<extended_property_definition, ‘thruster moulded form design parameter’, ‘moulded form characteristics definition’>] [CLASS<extended_property_definition, ‘moulded form characteristics definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
thruster_tunnel_diameter	value_representation_item.value_component	43	73, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘thruster moulded form design parameter’, ‘thruster tunnel diameter‘, positive_length_measure>
thruster_tunnel_min_length	value_representation_item.value_component	43	73, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘thruster moulded form design parameter’, ‘thruster tunnel min length‘, positive_length_measure>
thruster_tunnel_max_length	value_representation_item.value_component	43	73, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘thruster moulded form design parameter’, ‘thruster tunnel max length‘, positive_length_measure>
thruster_moulded_form_design_parameter to propeller_moulded_form_design_parameter (as thruster_propeller_parameter)	PATH		113	extended_property_definition <= property_definition <- property_definition_relationship.relating_property_definition property_definition_relationship {property_definition_relationship.name = ‘thruster propeller parameter’} property_definition_relationship property_definition_relationship.related_property_definition -> property_definition=> extended_property_definition {CLASS_ID<extended_property_definition, ‘propeller moulded form design parameter’>}
thruster_moulded_form_design_parameter to cartesian_point (as geometric_thruster_location)	PATH		73, 61	extended_property_definition <= PROP_DEF_TO_CART_POINT<‘thruster moulded form design parameter’, ‘geometric thruster location‘>
thruster_location	descriptive_representation_item	45	73	extended_property_definition <=

Application Element	AIM Element	Source	Rules	Reference Path
	.description			PROP_DEF_TO_DESC_REP_ITEM<‘thruster moulded form design parameter’, ‘thruster location‘> {(descriptive_representation_item.description = ‘bow‘) (descriptive_representation_item.description = ‘stern‘)}
from MOULDED_FORM_CHARACTERISTICS_DEFINITION				
description				*see supertype mappings*
moulded_form_outer_surface				*see supertype mappings*
moulded_form_displacement				*see supertype mappings*
thruster_moulded_form_design_parameter to moulded_form (as defined_for)				*see supertype mappings*
from DEFINITION				
thruster_moulded_form_design_parameter to global_id (as id)				*see supertype mappings*
thruster_moulded_form_design_parameter to unit (as local_units)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
DECK_MOULDED_FORM_DESIGN_PARAMETER	extended_property_definition	216	100	{[CLASS<extended_property_definition, ‘deck moulded form design parameter’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
camber	value_representation_item.value_component	43	69, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘deck moulded form design parameter’, ‘camber‘, positive_length_measure>
sheer_at_AP	value_representation_item.value_component	43	69, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘deck moulded form design parameter’, ‘sheer at AP‘, positive_length_measure>

Application Element	AIM Element	Source	Rules	Reference Path
sheer_at_FP	value_representation_item.value_component	43	69, 61	extended_property_definition <= PROP_DEF_TO_VAL_REP_ITEM<‘deck moulded form design parameter’, ‘sheer at FP‘, positive_length_measure>
deck_moulded_form_design_parameter to moulded_form (as defined_for)	PATH		69, 61	extended_property_definition <= PROP_TO_PROD_DEF {CLASS_ID<product_definition, ‘moulded_form’>}
from DEFINITION				
deck_moulded_form_design_parameter to global_id (as id)	extended_property_definition.id	216	63, 54	
deck_moulded_form_design_parameter to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<‘deck moulded form design parameter’>
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>

Table 4 - Mapping table for hydrostatics UoF

Application Element	AIM Element	Source	Rules	Reference Path
HYDROSTATIC_DEFINITION	extended_property_definition	216		{[CLASS<extended_property_definition, ‘hydrostatic definition’, ‘design definition’>] [CLASS<extended_property_definition, ‘design definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
displacement_change_description	property_definition.description	41		
hydrostatic_definition to ship_moulded_form (as defined_for)	PATH			extended_property_definition <= PROP_TO_PROD_DEF {CLASS_ID<product_definition, ‘ship moulded_form’>}
hydrostatic_definition to hydrostatic_table (as representations)	PATH		26	extended_property_definition <= PROP_DEF_TO_REP {CLASS_ID<representation, ‘hydrostatic table’>}
hydrostatic_definition to displacement_operation (as displacement_changes)	PATH			extended_property_definition <= property_definition <- property_definition_relationship.related_property_definition property_definition_relationship {property_definition_relationship.name = ‘displacement change‘} property_definition_relationship.relating_property_definition -> property_definition {CLASS_HELP<property_definition> (group.name = ‘addition of moulded form’) (group.name = ‘subtraction of moulded form’) class}
from DEFINITION				
hydrostatic_definition to global_id (as id)	extended_property_definition.id	216	63, 55	
hydrostatic_definition to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<‘hydrostatic definition’>

Application Element	AIM Element	Source	Rules	Reference Path
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
DISPLACEMENT_OPERATION	property_definition	41		*see subtype mappings*
displacement_operation to moulded_form (as displacement_of_moulded_form_to_add_or_subtract)	PATH			PROP_TO_PROD_DEF {CLASS_ID<product_definition, 'moulded_form'>}
ADDITION_OF_MOULDED_FORM	property_definition	41		{CLASS_ID<property_definition, 'addition of moulded_form'>}
from DISPLACEMENT_OPERATION				
addition_of_moulded_form to moulded_form (as displacement_of_moulded_form_to_add_or_subtract)				*see supertype mappings*
SUBTRACTION_OF_MOULDED_FORM	property_definition	41		{CLASS_ID<property_definition, 'subtraction of moulded_form'>}
from DISPLACEMENT_OPERATION				
subtraction_of_moulded_form to moulded_form (as displacement_of_moulded_form_to_add_or_subtract)				*see supertype mappings*
HYDROSTATIC_TABLE	representation	43		{CLASS_ID<representation, 'hydrostatic table'>}
mean_shell_thickness	value_representation_item.value_component	43	21, 61	REP_TO_VAL_REP_ITEM<'mean shell thickness', positive_length_measure>
hydrostatic_table to hydrostatic_property (as properties_in_the_hydrostatic_table) <LIST[1:?]	PATH		29	representation representation.items [i] -> representation_item => compound_representation_item {CLASS_ID<compound_representation_item, 'hydrostatic property'>}
hydrostatic_table to hydrostatic_properties_for_constant_floatin	PATH		28	representation representation.items [i] ->

Application Element	AIM Element	Source	Rules	Reference Path
g_position (as items)				representation_item => compound_representation_item {CLASS_ID<compound_representation_item, 'hydrostatic properties for constant floating position'>}
from REPRESENTATION				
name	represenation.name	43		
hydrostatic_table to representation_context (as representation_context)	PATH			represenation representation.context_of_items -> representation_context
HYDROSTATIC_PROPERTY	compound_representation_item	43		{CLASS_ID<compound_representation_item, 'hydrostatic property'>}
property_type	descriptive_representation_item	45	8	COMPOUND<'property type'> descriptive_representation_item {(descriptive_representation_item.description = 'waterplane area') (descriptive_representation_item.description = 'midship section area') (descriptive_representation_item.description = 'wetted surface area') (descriptive_representation_item.description = 'moment for unit change of trim') (descriptive_representation_item.description = 'longitudinal second moment of area of waterplane') (descriptive_representation_item.description = 'longitudinal metacentric height') (descriptive_representation_item.description = 'transverse second moment of area of waterplane') (descriptive_representation_item.description = 'transverse metacentric height') (descriptive_representation_item.description = 'centre of buoyancy') (descriptive_representation_item.description = 'centre of flotation') (descriptive_representation_item.description = 'centre of lateral resistance')}
property_unit	PATH			compound_representation_item <= representation_item <- represenation.items[i] represenation

Application Element	AIM Element	Source	Rules	Reference Path
				representation.context_of_items -> representation_context => global_unit_assigned_context global_unit_assigned_context.units
HYDROSTATIC_PROPERTIES_FOR_CONSTANT_FLOATING_POSITION	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘hydrostatic properties for constant floating position’>}
hydrostatic_properties_for_constant_floating_position to floating_position (as definition_of_floating_position)	PATH		16	COMPOUND<‘definition of floating position‘> compound_representation_item {CLASS_ID<compound_representation_item, ‘floating position’>}
hydrostatic_properties_for_constant_floating_position to hydrostatic_property_value (as hydrostatic_property_values)	PATH		34	COMPOUND<‘hydrostatic property value‘> (geometric_representation_item => point => cartesian_point) (value_representation_item)
hydrostatic_properties_for_constant_floating_position to hydrostatic_table (as related_hydrostatic_table)	PATH			compound_representation_item <= representation_item <- representation.items [i] representation {CLASS_ID<representation, ‘hydrostatic table’>}
from REPRESENTATION_ITEM				
name	representation_item.name	43		
FLOATING_POSITION	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘floating position’>}
moulded_form_displacement	value_representation_item.value_component	43	7, 61	COMPOUND<‘moulded form displacement‘> value_representation_item {value_representation_item.value_component -> measure_value = volume_measure}
draught_at_amidships	value_representation_item.value_component	43	7, 61	COMPOUND<‘draught at amidships‘> value_representation_item {value_representation_item.value_component -> measure_value = positive_length_measure}
length_of_waterline	value_representation_item.value_component	43	7, 61	COMPOUND<‘length of waterline‘> value_representation_item {value_representation_item.value_component -> measure_value = positive_length_measure}

Application Element	AIM Element	Source	Rules	Reference Path
breadth_of_waterline	value_representation_item.value_component	43	7, 61	COMPOUND<‘breadth of waterline‘> value_representation_item {value_representation_item.value_component -> measure_value = positive_length_measure}
angle_of_trim	value_representation_item.value_component	43	7, 61	COMPOUND<‘angle of trim‘> value_representation_item {value_representation_item.value_component -> measure_value = plane_angle_measure}
angle_of_heel	value_representation_item.value_component	43	7, 61	COMPOUND<‘angle of heel‘> value_representation_item {value_representation_item.value_component -> measure_value = plane_angle_measure}
HYDROSTATIC_PROPERTY_VALUE	(cartesian_point) (value_representation_item.value_component)	42 43		*see subtype mappings*
HYDROSTATIC_POSITION_VALUE	cartesian_point	42		
hydrostatic_position_value to cartesian_point (as position)	IDENTICAL MAPPING			
HYDROSTATIC_SCALAR_VALUE	value_representation_item.value_component	43		{value_representation_item value_representation_item.value_component -> measure_value = ratio_measure}
scalar	IDENTICAL MAPPING			
STABILITY_DEFINITION	extended_property_definition	216		{[CLASS<extended_property_definition, ‘stability definition’, ‘design definition’>] [CLASS<extended_property_definition, ‘design definition’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
intact_stability_definition to ship_moulded_form (as defined_for)	PATH			extended_property_definition <= PROP_TO_PROD_DEF {CLASS_ID<product_definition, ‘ship moulded_form’>}

Application Element	AIM Element	Source	Rules	Reference Path
stability_definition to stability_table (as representations)	PATH		27	extended_property_definition <= PROP_DEF_TO_REP {CLASS_ID<representation, ‘stability table’>}
from DEFINITION				
stability_definition to global_id (as id)	extended_property_definition.id	216	63, 60	
stability_definition to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<‘hydrostatic definition’>
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
STABILITY_TABLE	representation	41		{CLASS_ID<representation, ‘stability table’>}
mean_shell_thickness	value_representation_item.value_component	43	25, 61	REP_TO_VAL_REP_ITEM<‘mean shell thickness‘, positive_length_measure>
stability_table to stability_properties_for_one_floating_position (as items)	PATH		33	representation representation.items [i] -> representation_item => compound_representation_item {CLASS_ID<compound_representation_item, ‘stability properties for one floating position’>}
from REPRESENTATION				
name	represenation.name	43		
stability_table to representation_context (as representation_context)	PATH			represenation representation.context_of_items -> representation_context
STABILITY_PROPERTIES_FOR_ONE_FLOATING_POSITION	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘stability properties for one floating position’>}
stability_properties_for_one_floating_position to cartesian_point (as centre_of_gravity_above_keel)	PATH		35, 61	COMPOUND<‘centre of gravity above keel‘> geometric_representation_item => point => cartesian_point
stability_properties_for_one_floating_position to stability_property (as	PATH		20	COMPOUND<‘stability properties for different angles of heel‘> compound_representation_item {CLASS_ID<compound_representation_item, ‘stability property’>}

Application Element	AIM Element	Source	Rules	Reference Path
stability_properties_for_different_angles_of_heel)				
stability_properties_for_one_floating_position to floating_position (as definition_of_starting_floating_position)	PATH		16	COMPOUND<‘definition of starting floating position‘> compound_representation_item {CLASS_ID<compound_representation_item, ‘floating position’>}
stability_properties_for_constant_floating_position to stability_table (as related_stability_table)	PATH			compound_representation_item <= representation_item <- representation.items [i] representation {CLASS_ID<representation, ‘stability table’>}
from REPRESENTATION_ITEM				
name	representation_item.name	43		
STABILITY_PROPERTY	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘stability property’>}
angle_of_heel	value_representation_item.value_component	43	15, 61	COMPOUND<‘angle of heel‘> value_representation_item {value_representation_item.value_component -> measure_value = plane_angle_measure}
righting_arm	value_representation_item.value_component	43	15, 61	COMPOUND<‘righting arm‘> value_representation_item {value_representation_item.value_component -> measure_value = positive_length_measure}
stability_property to cartesian_point (as centre_of_buoyancy)	PATH		15, 61	COMPOUND<‘centre of buoyancy‘> geometric_representation_item => point => cartesian_point

Table 5 - Mapping table for location_concepts UoF

Application Element	AIM element	Source	Rules	Reference Path
GLOBAL_AXIS_PLACEMENT	[product_definition] [axis2_placement_3d]	41 42	3, 77, 61	{[CLASS<product_definition, ‘global axis placement’, ‘general characteristics definition’>] [CLASS<product_definition, ‘general characteristics definition’, ‘definition’>] [CLASS<product_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<product_definition, ‘versionable object’>]} PROD_DEF_TO_REP<‘global axis placement’> representation {representation.name = ‘global axis representation‘} representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context {geometric_representation_context.coordinate_space_dimension = 3} REP_ITEM<’global axes and origin’> geometric_representation_item => placement => axis2_placement_3d
after_perpendicular_offset	value_representation_item.value_component	43	77, 61	PROD_DEF_TO_REP<‘global axis placement’> representation {representation.name = ‘global axis representation‘} representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context REP_TO_VAL_REP_ITEM<‘after perpendicular offset‘, length_measure>
orientation	descriptive_representation_item.description	45	77, 61	PROD_DEF_TO_REP<‘global axis placement’> representation {representation.name = ‘global axis representation‘} representation.context_of_items

Application Element	AIM element	Source	Rules	Reference Path
				representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context REP_ITEM<‘orientation’> descriptive_representation_item descriptive_representation_item.description {(descriptive_representation_item.description = ‘forward poining‘) (descriptive_representation_item.description = ‘aft pointing‘)}
from GENERAL_CHARACTERISTICS_DEFINITION				
global_axis_placement to ship (as defined_for)	PATH			PROD_DEF_PRODUCT {CLASS_ID<product, ‘ship’>}
from DEFINITION				
global_axis_placement to global_id (as id)	product_definition.id	41	63, 47	
global_axis_placement to unit (as local_units)	PATH			PROD_DEF_TO_UNITS<‘global axis placement’>
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<product_definition>
LOCAL_CO_ORDINATE_SYSTEM	[extended_property_definition] [axis2_placement_3d]	216 42	108, 79, 61	{[CLASS<extended_property_definition, ‘local co-ordinate system’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]} PROP_DEF_REP_HELP<‘local co-ordinate system’> representation {representation.name = ‘local axis representation‘} representation.context_of_items representation_context => {representation_context.context_type = ‘local co-ordinate space’} geometric_representation_context {geometric_representation_context.coordinate_space_dimension = 3}

Application Element	AIM element	Source	Rules	Reference Path
				REP_ITEM<'local axes and origin'> geometric_representation_item => placement => axis2_placement_3d
parent	PATH		111	axis2_placement_3d <= placement <= geometric_representation_item <= representation_item <- item_defined_transformation.transform_item_1 item_defined_transformation {item_defined_transformation.name = 'transformation to parent'} item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => placement => axis2_placement_3d {CLASS_HELP<axis2_placement_3d> (group.name = 'local co-ordinate system') (group.name = 'local co-ordinate system with position reference') (group.name = 'global axis placement') class}
from DEFINITION				
local_co_ordinate_system to global_id (as id)	extended_property_definition.id	216	63, 56	
local_co_ordinate_system to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<'local co-ordinate system'>
local_co_ordinate_system to definable_object (as defined_for)	PATH			extended_property_definition <= PROP_TO_PROD_DEF
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
from AXIS2_PLACEMENT_3D				
local_co_ordinate_system to direction	PATH			axis2_placement_3d

Application Element	AIM element	Source	Rules	Reference Path
(as axis)				axis2_placement_3d.axis -> direction
local_co_ordinate_system to direction (as ref_direction)	PATH			axis2_placement_3d axis2_placement_3d.ref_direction -> direction
from PLACEMENT				
local_co_ordinate_system to cartesian_point (as location)	PATH			axis2_placement_3d <= placement placement.location -> cartesian_point
from REPRESENTATION_ITEM				
name	property_definition.name	41		
LOCAL_CO_ORDINATE_SYSTEM_WITH_POSITION_REFERENCE	[extended_property_definition] [axis2_placement_3d]	216 42	107, 78, 61	{[CLASS<extended_property_definition, ‘local co-ordinate system with position reference’, ‘local co-ordinate system’>] [CLASS<extended_property_definition, ‘local co-ordinate system’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]} PROP_DEF_REP_HELP<‘local co-ordinate system with position reference’> representation {representation.name = ‘local axis with position reference representation‘} representation.context_of_items representation_context => {representation_context.context_type = ‘local co-ordinate space’} geometric_representation_context {geometric_representation_context.coordinate_space_dimension = 3} REP_ITEM<’local axes and origin’> geometric_representation_item => placement => axis2_placement_3d
local_co_ordinate_system_with_position_re	PATH			axis2_placement_3d <=

Application Element	AIM element	Source	Rules	Reference Path
ference to cartesian_point (as location)				placement placement.location -> cartesian_point
longitudinal_ref	(value_representation_item.value_component) (compound_representation_item)	43 43		extended_property_definition<= PROP_DEF_REP_HELP<‘local co-ordinate system with position reference’> representation {representation.name = ‘local axis with position reference representation‘} representation.items [i] -> representation_item (representation_item => compound_representation_item {CLASS_HELP<compound_representation_item> (group.name = ‘longitudinal position’) (group.name = ‘spacing position with offset’) class}) (representation_item => {representation_item.name = ‘longitudinal ref’} value_representation_item {value_representation_item.value_component -> measure_value = length_measure})
transversal_ref	(value_representation_item.value_component) (compound_representation_item)	43 43		extended_property_definition<= PROP_DEF_REP_HELP<‘local co-ordinate system with position reference’> representation {representation.name = ‘local axis with position reference representation‘} representation.items [i] -> representation_item (representation_item => compound_representation_item {CLASS_HELP<compound_representation_item> (group.name = ‘transversal position’)

Application Element	AIM element	Source	Rules	Reference Path
				(group.name = ‘spacing position with offset’) class}) (representation_item => {representation_item.name = ‘transversal ref’} value_representation_item {value_representation_item.value_component -> measure_value = length_measure})
vertical_ref	(value_representation_item.value_component) (compound_representation_item)	43 43		extended_property_definition<= PROP_DEF_REP_HELP<‘local co-ordinate system with position reference’> representation {representation.name = ‘local axis with position reference representation‘} representation.items [i] -> representation_item (representation_item => compound_representation_item {CLASS_HELP<compound_representation_item> (group.name = ‘vertical position’) (group.name = ‘spacing position with offset’) class}) (representation_item => {representation_item.name = ‘vertical ref’} value_representation_item {value_representation_item.value_component -> measure_value = length_measure})
from LOCAL_CO_ORDINATE_SYSTEM				
parent				*see supertype mappings*
from DEFINITION				
local_co_ordinate_system_with_position_reference to global_id (as id)				*see supertype mappings*
local_co_ordinate_system_with_position_reference to unit				*see supertype mappings*

Application Element	AIM element	Source	Rules	Reference Path
(as local_units)				
local_co_ordinate_system_with_position_reference to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
from AXIS2_PLACEMENT_3D				
local_co_ordinate_system_with_position_reference to direction (as axis)				*see supertype mappings*
local_co_ordinate_system_with_position_reference to direction (as ref_direction)				*see supertype mappings*
from REPRESENTATION_ITEM				
name				*see supertype mappings*
SPACING_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
description	property_definition.description	43		
name	property_definition.name	43		
spacing_table to spacing_position (as spacing_table_representations)	PATH			extended_property_definition<= PROP_DEF_TO_REP representation.items [i] -> representation_item => compound_representation_item {CLASS_HELP<compound_representation_item> (group.name = ‘longitudinal position’) (group.name = ‘transversal position’) (group.name = ‘vertical position’) (group.name = ‘spacing position with offset’) class}

Application Element	AIM element	Source	Rules	Reference Path
from DEFINITION				
spacing_table to global_id (as id)	extended_property_definition.id	216	63, 59	
spacing_table to unit (as local_units)	PATH			extended_property_definition <= PROP_DEF_TO_UNITS<‘spacing table’>
spacing_table to definable_object (as defined_for)	PATH			extended_property_definition <= property_definition property_defintion.definition -> characterized_definition characterized_definition = characterized_product_definition (characterized_product_definition = product_definition) (characterized_product_definition = product_definition product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product)
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<extended_property_definition>
LONGITUDINAL_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘longitudinal table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
longitudinal_table to longitudinal_position (as spacing_table_representations)	PATH			extended_property_definition<= PROP_DEF_TO_REP representation.items [i] -> representation_item => compound_representation_item {CLASS_ID<compound_representation_item, ‘longitudinal position’>}
from SPACING_TABLE				
description				*see supertype mappings*

Application Element	AIM element	Source	Rules	Reference Path
name				*see supertype mappings*
from DEFINITION				
longitudinal_table to global_id (as id)				*see supertype mappings*
longitudinal_table to unit (as local_units)				*see supertype mappings*
longitudinal_table to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
STATION_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘station table’, ‘longitudinal table’>] [CLASS<extended_property_definition, ‘longitudinal table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
from LONGITUDINAL_TABLE				
station_table to longitudinal_position (as spacing_table_representations)				*see supertype mappings*
from SPACING_TABLE				
description				*see supertype mappings*
name				*see supertype mappings*
from DEFINITION				
station_table to global_id (as id)				*see supertype mappings*
station_table to unit (as local_units)				*see supertype mappings*
station_table to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*

Application Element	AIM element	Source	Rules	Reference Path
FRAME_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘frame table’, ‘longitudinal table’>] [CLASS<extended_property_definition, ‘longitudinal table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
from LONGITUDINAL_TABLE				
frame_table to longitudinal_position (as spacing_table_representations)				*see supertype mappings*
from SPACING_TABLE				
description				*see supertype mappings*
name				*see supertype mappings*
from DEFINITION				
frame_table to global_id (as id)				*see supertype mappings*
frame_table to unit (as local_units)				*see supertype mappings*
frame_table to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
TRANSVERSAL_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘transversal table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
transversal_table to transversal_position (as spacing_table_representations)	PATH			extended_property_definition<= PROP_DEF_TO_REP representation.items [i] -> representation_item => compound_representation_item

Application Element	AIM element	Source	Rules	Reference Path
				{CLASS_ID<compound_representation_item, ‘transversal position’>}
from SPACING_TABLE				
description				*see supertype mappings*
name				*see supertype mappings*
from DEFINITION				
transversal_table to global_id (as id)				*see supertype mappings*
transversal_table to unit (as local_units)				*see supertype mappings*
transversal_table to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
BUTTOCK_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘buttock table’, ‘transversal table’>] [CLASS<extended_property_definition, ‘transversal table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
from TRANSVERSAL_TABLE				
buttock_table to transversal_position (as spacing_table_representations)				*see supertype mappings*
from SPACING_TABLE				
description				*see supertype mappings*
name				*see supertype mappings*
from DEFINITION				
buttock_table to global_id (as id)				*see supertype mappings*
buttock_table to unit (as local_units)				*see supertype mappings*
buttock_table to definable_object				*see supertype mappings*

Application Element	AIM element	Source	Rules	Reference Path
(as defined_for)				
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
VERTICAL_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘vertical table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>] ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
vertical_table to vertical_position (as spacing_table_representations) <LIST[0:?]	PATH			extended_property_definition<= PROP_DEF_TO_REP representation.items [i] -> representation_item => compound_representation_item {CLASS_ID<compound_representation_item, ‘vertical position’>}
from SPACING_TABLE				
description				*see supertype mappings*
name				*see supertype mappings*
from DEFINITION				
vertical_table to global_id (as id)				*see supertype mappings*
vertical_table to unit (as local_units)				*see supertype mappings*
vertical_table to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
WATERLINE_TABLE	extended_property_definition	216		{[CLASS<extended_property_definition, ‘waterline table’, ‘vertical table’>] [CLASS<extended_property_definition, ‘vertical table’, ‘spacing table’>] [CLASS<extended_property_definition, ‘spacing table’, ‘definition’>] [CLASS<extended_property_definition, ‘definition’, ‘versionable object’>]

Application Element	AIM element	Source	Rules	Reference Path
				ROOT_CLASS<extended_property_definition, ‘versionable object’>]}
from VERTICAL_TABLE				
vertical_table to vertical_position (as spacing_table_representations)				*see supertype mappings*
from SPACING_TABLE				
description				*see supertype mappings*
name				*see supertype mappings*
from DEFINITION				
vertical_table to global_id (as id)				*see supertype mappings*
vertical_table to unit (as local_units)				*see supertype mappings*
vertical_table to definable_object (as defined_for)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id				*see supertype mappings*
SPACING_POSITION	compound_representation_item	43		{ROOT_CLASS<compound_representation_item, ‘spacing position’>} {compound_representation_item <- representation.items[i] representation representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context}
name	compound_representation_item .name	43		
position_number	value_representation_item.value_component	43	14, 61	COMPOUND<‘position number‘> value_representation_item {value_representation_item.value_component -> measure_value = count_measure}
position	value_representation_item.value_component	43	14, 61	COMPOUND<‘position‘> value_representation_item {value_representation_item.value_component ->

Application Element	AIM element	Source	Rules	Reference Path
				measure_value = length_measure}
LONGITUDINAL_POSITION	compound_representation_item	43		{CLASS<compound_representation_item, ‘longitudinal position’, ‘spacing position’> ROOT_CLASS<compound_representation_item, ‘spacing position’>} {compound_representation_item <- representation.items[i] representation representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context}
from SPACING_POSITION				
name				*see supertype mappings*
position_number				*see supertype mappings*
position				*see supertype mappings*
TRANSVERSAL_POSITION	compound_representation_item	43		{CLASS<compound_representation_item, ‘transversal position’, ‘spacing position’> ROOT_CLASS<compound_representation_item, ‘spacing position’>} {compound_representation_item <- representation.items[i] representation representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context}
from SPACING_POSITION				
name				*see supertype mappings*
position_number				*see supertype mappings*
position				*see supertype mappings*
VERTICAL_POSITION	compound_representation_item	43		{CLASS<compound_representation_item, ‘vertical position’, ‘spacing position’> ROOT_CLASS<compound_representation_item, ‘spacing position’>} {compound_representation_item <- representation.items[i]

Application Element	AIM element	Source	Rules	Reference Path
				representation representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context}
from SPACING_POSITION				
name				*see supertype mappings*
position_number				*see supertype mappings*
position				*see supertype mappings*
SPACING_POSITION_WITH_OFFSET	compound_representation_item	43		{CLASS<compound_representation_item, ‘spacing position with offset’, ‘spacing position’> ROOT_CLASS<compound_representation_item, ‘spacing position’>} {compound_representation_item <- representation.items[i] representation representation.context_of_items representation_context => {representation_context.context_type = ‘global co-ordinate space’} geometric_representation_context}
offset	value_representation_item.value_component	43	13, 61	COMPOUND<‘offset‘> value_representation_item {value_representation_item.value_component -> measure_value = length_measure}
spacing_position_reference to spacing_position (as relating_spacing_position)	PATH		19	COMPOUND<‘relating spacing position‘> {CLASS_HELP<compound_representation_item> (group.name = ‘longitudinal position’) (group.name = ‘transversal position’) (group.name = ‘vertical position’) class}
position	value_representation_item.value_component	43		COMPOUND<‘position‘> value_representation_item {value_representation_item.value_component -> measure_value = length_measure}
from SPACING_POSITION				

Application Element	AIM element	Source	Rules	Reference Path
name				*see supertype mappings*
position_number				*see supertype mappings*

Table 6 - Mapping table for offset_table_representations UoF

Application Element	AIM Element	Source	Rules	Reference Path
OFFSET_TABLE_SHAPE_REPRESENTATION	shape_representation	41		{CLASS_ID<shape_representation, ‘offset table shape representation’>}
offset_table_shape_representation to offset_point_table_model (as items)	PATH		32	shape_representation <= representation representation.items[i] -> representation_item => compound_representation_item {CLASS_ID<compound_representation_item, ‘offset point table model’>}
from MOULDED_FORM_SHAPE_REPRESENTATION				
moulded_form_representation_id				*see supertype mappings*
offset_table_shape_representation to symmetry (as moulded_form_symmetry)				*see supertype mappings*
from REPRESENTATION				
name				*see supertype mappings*
offset_table_shape_representation to representation_context (as representation_context)				*see supertype mappings*
OFFSET_POINT_TABLE_MODEL	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘offset point table model’>}
offset_point_table_type	descriptive_representation_item .description	45	12	COMPOUND<‘offset point table type‘> descriptive_representation_item {(descriptive_representation_item.name = ‘waterline table’) (descriptive_representation_item.name = ‘station table’) (descriptive_representation_item.name = ‘buttock table’) (descriptive_representation_item.name = ‘user defined table’)}
offset_point_table_model to section_of_offset_point_table	PATH		30	COMPOUND<‘section of offset point table‘> compound_representation_item

Application Element	AIM Element	Source	Rules	Reference Path
(as offset_point_table_sections)				{CLASS_ID<compound_representation_item, ‘section of offset point table’>}
from REPRESENTATION_ITEM				
name	representation_item.name	43		
SECTION_OF_OFFSET_POINT_TABLE	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘section of offset point table’>}
section_identifier	identification_assignment.assigned_id	41	89	HAS_ID_1_ROLE<compound_representation_item, ‘section identifier’>
section_of_offset_point_table to ship_point (as section_points)	PATH		31	COMPOUND<‘section point‘> compound_representation_item {CLASS_ID<compound_representation_item, ‘ship point’>}
section_of_offset_point_table to offset_point_table_model (as for_table	PATH			compound_representation_item <= representation_item <- list_representation_item[i] compound_item_definition = list_representation_item <- compound_representation_item.item_element compound_representation_item {CLASS_ID<compound_representation_item, ‘offset point table model’>}

Table 7 - Mapping table for wireframe_representations UoF

Application Element	AIM Element	Source	Rules	Reference Path
WIREFRAME_SHAPE_REPRESENTATION	shell_based_wireframe_shape_representation	502		
wireframe_shape_representation to shell_based_wireframe_model (as items)	PATH			shell_based_wireframe_shape_representation <= shape_representation <= representation representation items[i] -> shell_based_wireframe_model
from MOULDED_FORM_SHAPE_REPRESENTATION				
moulded_form_representation_id				*see supertype mappings*
wireframe_shape_representation to symmetry (as moulded_form_symmetry)				*see supertype mappings*
from REPRESENTATION				
name				*see supertype mappings*
wireframe_shape_representation to representation_context (as representation_context)				*see supertype mappings*
SHIP_MOULDED_FORM_WIREFRAME_MODEL	shape_representation	41	24	{CLASS_ID<shape_representation, 'ship moulded form wireframe model'>} {shape_representation <= representation representation.items[i] -> representation_item => representation_item.name = 'wireframe' topological_representation_item => wire_shell}
ship_moulded_form_wireframe_model to ship_curve (as mesh_definition_ship_curves)	PATH		36	shape_representation <= representation representation.items[i] ->

Application Element	AIM Element	Source	Rules	Reference Path
				representation_item => {representation_item.name = ‘mesh definition ship curve’} compound_representation_item {CLASS_ID<compound_representation_item, ‘ship curve’>}
from SHELL_BASED_WIREFRAME_MODEL				
ship_moulded_form_wireframe_model to shell (as sbwm_boundary)				*see supertype mappings*
from REPRESENTATION_ITEM				
name				*see supertype mappings*
SHIP_CURVE_SEGMENT	[edge_curve] [compound_representation_item]	42 42		{CLASS_ID<edge_curve, ‘ship curve segment’> CLASS_ID<compound_representation_item, ‘ship curve segment’>}
ship_curve_segment to ship_curve (as part_of_ship_curve)	PATH		17	COMPOUND<‘ship curve segment‘> compound_representation_item {CLASS_ID<compound_representation_item, ‘ship curve’>}
from EDGE_CURVE				
ship_curve_segment to curve (as edge_geometry)	PATH			edge_curve edge_curve.edge_geometry -> curve
same_sense	edge_curve.same_sense	42		
from EDGE				
ship_curve_segment to vertex (as edge_start)	PATH			edge edge.edge_start -> vertex
ship_curve_segment to vertex (as edge_end)	PATH			edge edge.edge_end -> vertex
from REPRESENTATION_ITEM				
name	representation_item.name	43		
KNOT	[vertex_point] [compound_representation_ite	42 42		{CLASS_ID<vertex_point, ‘knot’> CLASS_ID<compound_representation_item, ‘knot’>}

Application Element	AIM Element	Source	Rules	Reference Path
	m]			
knot to ship_curve (as intersecting_ship_curves)	PATH		37	COMPOUND<‘intersecting ship curve‘> compound_representation_item {CLASS_ID<compound_representation_item, ‘ship curve’>}
knot to vector (as tangent_information)	PATH			COMPOUND<‘tangent vector‘> geometric_representation_item => vector
from VERTEX_POINT				
knot to point (as vertex_geometry)	PATH			vertex_point vertex_point.vertex_geometry -> point
from REPRESENTATION_ITEM				
name	representation_item.name	43		

Table 8 - Mapping table for surface_representations UoF

Application Element	AIM Element	Source	Rules	Reference Path
SURFACE_SHAPE_REPRESENTATION	non_manifold_surface_shape_representation	508		
surface_shape_representation to face_based_surface_model (as items)	PATH			non_manifold_surface_shape_representation <= shape_representation <= representation representation items[i] -> face_based_surface_model
from MOULDED_FORM_SHAPE_REPRESENTATION				
moulded_form_representation_id				*see supertype mappings*
surface_shape_representation to symmetry (as moulded_form_symmetry)				*see supertype mappings*
from REPRESENTATION				
name				*see supertype mappings*
surface_shape_representation to representation_context (as representation_context)				*see supertype mappings*

Table 9 - Mapping table for basic_geometry UoF

Application Element	AIM Element	Source	Rules	Reference Path
SHIP_POINT	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘ship point’>}
point_class	representation_item.name	43		compound_representation_item <= representation_item {(representation_item.name = ‘ordinary’) (representation_item.name =‘tangent’) (representation_item.name =‘knuckle’) (representation_item.name =‘unspecified’)}
ship_point to point (as point_shape)	PATH		9, 61	COMPOUND<‘point shape‘> geometric_representation_item => point
from REPRESENTATION_ITEM				
name	representation_item.name	43		{(representation_item.name = ‘ordinary’) (representation_item.name =‘tangent’) (representation_item.name =‘knuckle’) (representation_item.name =‘unspecified’)}
SHIP_CURVE	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘ship curve’>}
curve_class	representation_item.name	43		compound_representation_item <= representation_item {(representation_item.name = ‘buttock line’) (representation_item.name = ‘centreline’) (representation_item.name = ‘camber') (representation_item.name = ‘sheer’) (representation_item.name = ‘flat of bottom’) (representation_item.name = ‘flat of side’) (representation_item.name = ‘intersection line’) (representation_item.name = ‘station line’) (representation_item.name = ‘bounding line’) (representation_item.name = ‘rise of floor’) (representation_item.name = ‘trace line’) (representation_item.name = ‘waterline’) (representation_item.name = ‘unspecified’)}

Application Element	AIM Element	Source	Rules	Reference Path
side_condition	descriptive_representation_item .description	45	10	COMPOUND<‘side condition‘> descriptive_representation_item {(descriptive_representation_item.description = ‘knuckle’) (descriptive_representation_item.description = ‘smooth’) (descriptive_representation_item.description = ‘tangent’) (descriptive_representation_item.description = ‘unspecified’)}
ship_curve to curve (as curve_shape)	PATH		10, 61	COMPOUND<‘curve shape‘> geometric_representation_item => curve
from REPRESENTATION_ITEM				
name	representation_item.name	43		{(representation_item.name = ‘buttock line’) (representation_item.name = ‘centreline’) (representation_item.name = ‘camber') (representation_item.name = ‘sheer’) (representation_item.name = ‘flat of bottom’) (representation_item.name = ‘flat of side’) (representation_item.name = ‘intersection line’) (representation_item.name = ‘station line’) (representation_item.name = ‘bounding line’) (representation_item.name = ‘rise of floor’) (representation_item.name = ‘trace line’) (representation_item.name = ‘waterline’) (representation_item.name = ‘unspecified’)}
SHIP_CURVE_WITH_SPACING_POSITION	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘ship curve with spacing position’>}
ship_curve_with_spacing_position to spacing_position (as location)	PATH		18	COMPOUND<‘spacing position‘> compound_representation_item {CLASS_HELP<compound_representation_item> (group.name = ‘longitudinal position’) (group.name = ‘transversal’position’) (group.name = ‘vertical position’) (group.name = ‘spacing position with offset’) class}
from SHIP_CURVE				

Application Element	AIM Element	Source	Rules	Reference Path
curve_class				*see supertype mappings*
side_condition				*see supertype mappings*
ship_curve to curve (as curve_shape)				*see supertype mappings*
from REPRESENTATION_ITEM				
name	representation_item.description	43		
SHIP_SURFACE	compound_representation_item	43		{CLASS_ID<compound_representation_item, ‘ship surface’>}
surface_class	representation_item.name	43		compound_representation_item <= representation_item {(representation_item.name = ‘external surface’) (representation_item.name = ‘blending_surface’) (representation_item.name = ‘internal surface’)}
ship_surface to surface (as surface_shape)	PATH		11, 61	COMPOUND<‘surface shape‘> geometric_representation_item => surface
from REPRESENTATION_ITEM				
name	representation_item.name	43		{(representation_item.name = ‘external surface’) (representation_item.name = ‘blending_surface’) (representation_item.name = ‘internal surface’)}

Table 10 - Mapping table for items UoF

Application Element	AIM element	Source	Rules	Reference Path
DEFINABLE_OBJECT #1: ship #2: ship_moulded_form #3: moulded_form #4: moulded_form_relationship	#1: (product) #2: ([product_definition] [group]) #3: (product_definition) #4: (product_definition_relationship)	#1/ #2/ #3/ #4: 41		#1: {ROOT_CLASS<product, ‘definable object’>} #2/ #3: {ROOT_CLASS<product_definition, ‘definable object’>} #4: {ROOT_CLASS<product_definition_relationship, ‘definable object’>}
definable_object to definition (as definitions) #1: ship #2: ship_moulded_form #3: moulded_form #4: moulded_form_relationship	#1/ #2/ #3/ #4: PATH			#1: (product <- product_definition_formation.of_product product_definition_formation <- product_definition.formation product_definition) #2/ #3: (PROD_DEF_PROP_DEF) #4: (product_definition_relationship product_definition_relationship.related_product_definition ->PROD_DEF_PROP_DEF product_definition_relationship product_definition_relationship.relating_product_definition ->PROD_DEF_PROP_DEF)
definable_object to global_id (as id) #1: ship #2: ship_moulded_form #3: moulded_form	#1: (product.id) #2/ #3: (product_definition.id) #4: (product_definition_relationship.id)	#1/ #2/ #3/ #4: 41	#1: 63, 44 #2/ #3: 63, 46 #4: 63, 52	

Application Element	AIM element	Source	Rules	Reference Path
#4: moulded_form_relationship				
VERSIONABLE_OBJECT				*see subtype mappings*
version_id	applied_identification_assignment	41		*see subtype mappings*
ITEM #1: ship #2: ship_moulded_form #3: moulded_form	#1: (product) #2: ([product_definition] [group]) #3: (product_definition)	#1/ #2/ #3/: 41		
name #1: ship #2: ship_moulded_form #3: moulded_form	#1: (product.name) #2: (product_definition.name) #3: (product_definition.name)	#1/ #2/ #3/: 41		
description #1: ship #2: ship_moulded_form #3: moulded_form	#1: (product.description) #2: (product_definition.description) #3: (product_definition.description)	#1/ #2/ #3/: 41		
item to external_reference (as documentation) #1: ship #2: ship_moulded_form #3: moulded_form	#1/ #2/ #3/: PATH			#1: EXT_REFERENCE <product> #2/ #3: EXT_REFERENCE <product_definition>
item to ship (as ship_context) #1: ship (attribute does not exist)	#2/ #3/: PATH			#2/ #3/: PROD_DEF_PRODUCT {CLASS_ID<product, ‘ship’>}

Application Element	AIM element	Source	Rules	Reference Path
#2: ship_moulded_form #3: moulded_form				
from DEFINABLE_OBJECT				
item to global_id (as id)				*see supertype mappings*
item to definition (as definitions)				*see supertype mappings*
ITEM_RELATIONSHIP	product_definition_relationship	41		
contxt	product_definition_relationship .description	41		
item_relationship to external_instance_reference (as external_item_1)	PATH			*see subtype mappings*
item_relationship to external_instance_reference (as external_item_2)	PATH			*see subtype mappings*
item_relationship to item (as item_1)	PATH			*see subtype mappings*
item_relationship to item (as item_2)	PATH			*see subtype mappings*
from DEFINABLE_OBJECT				
item_relationship to global_id (as id)				*see supertype mappings*
item_relationship to definition (as definitions)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assig ned_id	41	62	VERSION_ID<product_definition_relationship>
ITEM_STRUCTURE	[product_definition] [group]	41 41		
item_structure to external_instance_reference	PATH			*see subtype mappings*

Application Element	AIM element	Source	Rules	Reference Path
(as external_items)				
item_structure to external_instance_reference (as external_relationships)	PATH			*see subtype mappings*
item_structure to item (as items)	PATH			*see subtype mappings*
item_structure to item_relationship (as relationships)	PATH			*see subtype mappings*
from DEFINABLE_OBJECT				
item_structure to global_id (as id)				*see supertype mappings*
item_structure to definition (as definitions)				*see supertype mappings*
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41	62	VERSION_ID<group>
SHIP	product	41		{[CLASS<product, ‘ship’, ‘item‘>] [CLASS<product, ‘item‘,‘definable object’>] [ROOT_CLASS <product, ‘definable object’>]}
units	PATH			product <- product_definition_formation.of_product product_definition_formation <- product_definition.formation PROD_DEF_TO_UNITS<’global units’>
ship to item (as ship_items)	PATH			product <- product_definition_formation.of_product product_definition_formation <- product_definition.formation product_definition
ITEM				
name				*see supertype mappings*
description				*see supertype mappings*
ship to external_reference (as documentation)				*see supertype mappings*

Application Element	AIM element	Source	Rules	Reference Path
ship to ship (as ship_context)				*see supertype mappings*
from DEFINABLE_OBJECT				
ship to global_id (as id)				*see supertype mappings*
ship to definition (as definitions)				*see supertype mappings*
GLOBAL_ID	identifier	41		
id	IDENTICAL MAPPING			
company_id	IDENTICAL MAPPING			
local_id	IDENTICAL MAPPING			

Table 11 - Mapping table for definitions UoF

Application Element	AIM element	Source	Rules	Reference Path
DEFINITION				*see subtype mappings*
local_units				*see subtype mappings*
definition to definable_object (as defined_for)				*see subtype mappings*
definition to global_id (as id)				*see subtype mappings*
from VERSIONABLE_OBJECT				
version_id				*see subtype mappings*
DEFINITION_RELATIONSHIP	property_definition_relationship	45		
description	property_definition_relationship.description	45		
definition_relationship to definition (as definition_2)	PATH			property_definition_relationship property_definition_relationship.related_property_definition -> property_definition
definition_relationship to definition (as definition_2)	PATH			property_definition_relationship property_definition_relationship.relating_property_definition -> property_definition
DESIGN_DEFINITION	extended_product_definition_shape	216		*see subtype mappings*
design_definition to representation (as representations)				*see subtype mappings*
from DEFINITION				
design_definition to global_id (as id)				*see subtype mappings*
design_definition ble to unit (as local_units)				*see subtype mappings*
design_definition to definable_object (as defined_for)				*see subtype mappings*
from VERSIONABLE_OBJECT				
version_id				*see subtype mappings*

Application Element	AIM element	Source	Rules	Reference Path
FUNCTIONAL_DEFINITION				*see subtype mappings*
user_def_function				*see subtype mappings*
functional_definition to unit (as local_units)				*see subtype mappings*
from DEFINITION				
functional_definition to global_id (as id)				*see subtype mappings*
functional_definition to definable_object (as defined_for)				*see subtype mappings*
from VERSIONABLE_OBJECT				
version_id				*see subtype mappings*
GENERAL_CHARACTERISTICS_DEFINITION	product_definition	41		*see subtype mappings*
general_characteristics_definition to ship (as defined_for)	PATH			PROD_DEF_PRODUCT {CLASS_ID<product, ‘ship’>}
from DEFINITION				
general_characteristics_definition to global_id (as id)				*see subtype mappings*
general_characteristics_definition to unit (as local_units)				*see subtype mappings*
from VERSIONABLE_OBJECT				
version_id				*see subtype mappings*

Table 12 - Mapping table for ship_measures UoF

Application Element	AIM element	Source	Rules	Reference Path
EXTENDED_CONVERSION_BASED_UNIT	conversion_based_unit	41		
name	conversion_based_unit.name	41		
extended_conversion_based_unit to extended_measure_with_unit (as conversion_factor)	PATH			conversion_based_unit conversion_based_unit.conversion_factor -> measure_with_unit
from NAMED_UNIT				
dimensions				
EXTENDED_MEASURE_WITH_UNIT	measure_with_unit	41		
unit_component	measure_with_unit.unit_component	41		
value_component	measure_with_unit.value_component	41		
SPEED_MEASURE_WITH_UNIT	measure_with_unit			{measure_with_unit classification_item = measure_with_unit classification_item <- applied_classification_assignment.items[i] applied_classification_assignment <= classification_assignment {classification_assignment.role -> classification_role classification_role.name = 'class membership'} classification_assignment classification_assignment.assigned_classification -> group => group.name = 'speed measure with unit' class}
SPEED_MEASURE	speed_measure	41		speed_measure = context_dependent_measure
SPEED_UNIT	derived_unit	41		derived_unit {derived_unit.name = 'speed unit' [derived_unit.elements[1] ->

Application Element	AIM element	Source	Rules	Reference Path
				derived_unit_element derived_unit_element.unit ->\ named_unit =>\ length_unit derived_unit_element.exponent ->\ derived_unit_element.exponent = 1.0] [derived_unit.elements [2] ->\ derived_unit_element derived_unit_element.unit ->\ named_unit <= time_unit derived_unit_element.exponent = -1.0]}

Table 13 - Mapping table for UoF external_references

Application Element	AIM element	Source	Rules	Reference Path
EXTERNAL_REFERENCE	applied_external_identification_assignment	216		applied_external_identification_assignment <= external_identification_assignment <= ID_ROLE<'external reference'>
description	identification_role.description	41		applied_external_identification_assignment <= external_identification_assignment <= identification_assignment identification_assignment.role -> identificaton_role identification_role.description
location	external_identification_assignment.source	41		applied_external_identification_assignment <= external_identification_assignment external_identification_assignment.source -> external_source {(CLASS_ID<external_source, 'universal resource locator'>) (CLASS_ID<external_source, 'address'>)}
UNIVERSAL_RESOURCE_LOCATOR	external_source	41		{ROOT_CLASS<external_source, 'universal resource locator'>}
location	external_source.source_id	41		external_source external_source.source_id -> source_item {source_item = identifier}
machine_address	external_source.source_id	41		external_source external_source.source_id -> source_item {source_item = identifier}
other_protocol_type	external_source.source_id	41		external_source external_source.source_id -> source_item {source_item = identifier}
port	external_source.source_id	41		external_source external_source.source_id -> source_item

Application Element	AIM element	Source	Rules	Reference Path
				{source_item = identifier}
protocol	external_source.source_id	41		external_source external_source.source_id -> source_item {source_item = identifier}
ADDRESS	external_source	41	114	{ROOT_CLASS<external_source, 'address'>}
internal_location	external_source.source_id	41		EXT_SRC_REL< 'internal location'>
street_number	external_source.source_id	41		EXT_SRC_REL< 'street number'>
street	external_source.source_id	41		EXT_SRC_REL< 'street'>
postal_box	external_source.source_id	41		EXT_SRC_REL< 'postal box'>
town	external_source.source_id	41		EXT_SRC_REL< 'town'>
region	external_source.source_id	41		EXT_SRC_REL< 'region'>
postal_code	external_source.source_id	41		EXT_SRC_REL< 'postal code'>
country	external_source.source_id	41		EXT_SRC_REL< 'country'>
facsimile_number	external_source.source_id	41		EXT_SRC_REL< 'facsimile number'>
telephone_number	external_source.source_id	41		EXT_SRC_REL< 'telephone number'>
electronic_mail_address	external_source.source_id	41		EXT_SRC_REL< 'electronic mail address'>
telex_number	external_source.source_id	41		EXT_SRC_REL< 'telex number'>
EXTERNAL_INSTANCE_REFERENCE	applied_external_identification_assignment	216		applied_external_identification_assignment <= external_identification_assignment <= ID_ROLE<'external instance reference'> external_identification_assignment external_identification_assignment.source -> ROOT_CLASS<external_source, 'instance address'>
schema_name	external_source.source_id	41		applied_external_identification_assignment <= external_identification_assignment external_identification_assignment.source -> EXT_SRC_REL< 'schema name' >
entity_type	external_source.source_id	41		applied_external_identification_assignment <= external_identification_assignment external_identification_assignment.source -> EXT_SRC_REL< 'entity type' >
EXTERNAL_INSTANCE_REFERENCE to	PATH	41		applied_external_identification_assignment <=

Application Element	AIM element	Source	Rules	Reference Path
GLOBAL_ID (as target_guid)				external_identification_assignment <= identification_assignment identification_assignment.assigned_id
DOCUMENT	document	41		{[CLASS<document, ‘document’, ‘versionable object’>] [ROOT_CLASS<document, ‘versionable object’>]}
#1: DOCUMENT to PERSON (as has_author) #2: DOCUMENT to ORGANISATION (as has_author) #3: DOCUMENT to PERSON_AND_ORGANIZATION (as has_author)	#1: person_assignment.assigned_person #2: organization_assignment.assigned_organization #3: person_and_organization_assignment.assigned_ person_and_organization	41		#1: PERS_ASSGN<document, ‘document author’> #2: ORG_ASSGN<document, ‘document author’> #3: PERS_ORG_ASSGN<document, ‘document author’>
has_title	document.name	41		
source_type	document_type	41		document document.kind -> document_type
summary	document.description	41		
DOCUMENT to DOCUMENT_USAGE_CONSTRAINT (as document_subset)	PATH			document <- document_usage_constraint.source document_usage_constraint
from VERSIONABLE_OBJECT				
version_id	*see supertype*			VERSION_ID< document >
DOCUMENT_REFERENCE	applied_document_reference	216		applied_document_reference <= document_reference {document_reference document_reference.role -> object_role object_role.name = 'document reference'}
document_reference to document_referent (as assigned_document)	PATH			applied_document_reference <= document_reference document_reference.assigned_document ->

Application Element	AIM element	Source	Rules	Reference Path
				(document) (document <- document_usage_constraint.source document_usage_constraint)
DOCUMENT_REFERENCE_WITH_ADDRESS	[applied_document_reference] [applied_external_identification_assignment]	216		{applied_document_reference <= document_reference {document_reference document_reference.role -> object_role object_role.name = 'document reference'} external_identification_item = document_reference external_identification_item <- applied_external_identification_assignment.items[i] applied_external_identification_assignment <= external_identification_assignment <= ID_ROLE<'external reference'>}
from DOCUMENT_REFERENCE				
DOCUMENT_REFERENCE_WITH_ADDRESS to DOCUMENT_REFERENT (as assigned_document)	PATH			*see supertype*
from EXTERNAL_REFERENCE				
description	*see supertype*			
location	*see supertype*			
DOCUMENT_USAGE_CONSTRAINT	document_usage_constraint	41		
element_name	document_usage_constraint.subject_element	41		
line_number	identification_assignment.assigned_id	41		document_usage_constraint HAS_ID< document_usage_constraint, 'line number'> identification_assignment.assigned_id
page	identification_assignment.assigned_id	41		document_usage_constraint HAS_ID< document_usage_constraint, 'page'> identification_assignment.assigned_id
paragraph	identification_assignment.assig	41		document_usage_constraint

Application Element	AIM element	Source	Rules	Reference Path
	ned_id			HAS_ID< document_usage_constraint, 'paragraph'> identification_assignment.assigned_id
section	identification_assignment.assigned_id	41		document_usage_constraint HAS_ID< document_usage_constraint, 'section'> identification_assignment.assigned_id
DOCUMENT_USAGE_CONSTRAINT to DOCUMENT (as source)	PATH			document_usage_constraint document_usage_constraint.source -> document

Table 14 - Mapping table for UoF configuration_management

Application Element	AIM element	Source	Rules	Reference Path
APPROVAL_EVENT	approval	41		
result	approval_status.name	41		approval approval.status -> approval_status approval_status.name {(approval_status.name = 'noted') (approval_status.name = 'unapproved') (approval_status.name = 'conditionally_approved') (approval_status.name = 'approved') (approval_status.name = 'rejected')}
user_defined_result	approval_status.name	41		approval approval.status -> approval_status approval_status.name
APPROVAL_EVENT to APPROVAL_HISTORY (as approval_reference)	PATH			approval group_item = approval group_item <- applied_group_assignment.items[i] applied_group_assignment <= group_assignment {group_assignment.role -> object_role [object_role.name = 'approval'] [object_role.description = '.UNUSED.']} group_assignment.assigned_group -> group
from EVENT				
description	approval.level	41		
APPROVAL_EVENT to PERSON_AND_ORGANISATION (as caused_by)	PATH			approval <- approval_person_organization.authorised_approval approval_person_organization

Application Element	AIM element	Source	Rules	Reference Path
				{approval_person_organization approval_person_organization.role ->approval_role approval_role.role='caused by'} approval_person_organization.person_organization -> person_organization_select = person_and_organization person_and_organization
APPROVAL_EVENT to AND_TIME (as caused_when)	PATH			approval <-approval_date_time.dated_approval approval_date_time approval_date_time.date_time -> date_time_select = date_and_time date_and_time
APPROVAL_HISTORY	group	41		{ROOT_CLASS<group, 'approval history'>}
APPROVAL_HISTORY to APPROVAL_EVENT (as approvals)	PATH			group <-GROUPS<approval, 'approvals'>
APPROVAL_HISTORY to DEFINITION (as subject)	PATH			group <-GROUPS<approval, 'approval'> <-(APPROVES<product_defnition, 'subject'> {CLASS_ID<product_definition, 'definition'}) (APPROVES<property_definition, 'subject'> {CLASS_ID<property_definition, 'definition'}) (APPROVES<representation, 'subject'> {CLASS_ID<representation, 'definition'})
REVISION	group	41		{[CLASS<group, 'revision', 'versionable object'>] [ROOT_CLASS<group, 'versionable object'>]}
name	group.name	41		
reason	group.description	41		
REVISION to VERSIONABLE_OBJECT (as members)	PATH			group <-(RELATE_GROUP_2_VO<representation, 'members'>) (RELATE_GROUP_2_VO<product_definition, 'members'>) (RELATE_GROUP_2_VO<property_definition, 'members'>) (RELATE_GROUP_2_VO<shape_aspect, 'members' >)

Application Element	AIM element	Source	Rules	Reference Path
				(RELATE_GROUP_2_VO<product_category, 'members'>) (RELATE_GROUP_2_VO<document, 'members' >) (RELATE_GROUP_2_VO<product_definition_relationship, 'members' >) (RELATE_GROUP_2_VO<shape_aspect_relationship, 'members' >)
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41		VERSION_ID< group >
REVISION_WITH_CONTEXT	group	41		{[CLASS<group, 'revision with context', 'revision'>] [CLASS<group, 'revision', 'versionable object'>] [ROOT_CLASS<group, 'versionable object'>]}
REVISION_WITH_CONTEXT to DEFINABLE_OBJECT (as context_of_revision)	PATH			group <- (RELATE_GROUP_2_DO< product, 'context of revision'>) (RELATE_GROUP_2_DO<product_definition, 'context of revision'>) (RELATE_GROUP_2_DO<product_definition_relationship, 'context of revision'>) (RELATE_GROUP_2_DO< shape_aspect, 'context of revision'>) (RELATE_GROUP_2_DO< shape_aspect_relationship, 'context of revision'>)
from REVISION				
name	*see supertype*			
reason	*see supertype*			
REVISION to VERSIONABLE_OBJECT (as members)	PATH			*see supertype*
from VERSIONABLE_OBJECT				
version_id	identification_assignment.assigned_id	41		VERSION_ID< group >
VERSION_HISTORY	group	41		{ROOT_CLASS<group, 'version history'>}
VERSION_HISTORY to VERSIONABLE_OBJECT (as current_version)	PATH			group <- (RELATE_GROUP_2_VO<representation, 'versions'>) (RELATE_GROUP_2_VO<product_definition, 'versions'>) (RELATE_GROUP_2_VO<property_definition, 'versions'>)

Application Element	AIM element	Source	Rules	Reference Path
				(RELATE_GROUP_2_VO<shape_aspect, 'versions'>) (RELATE_GROUP_2_VO<product_category, 'versions', >) (RELATE_GROUP_2_VO<document, 'versions'>) (RELATE_GROUP_2_VO<product_definition_relationship, 'versions'>) (RELATE_GROUP_2_VO<shape_aspect_relationship, 'versions'>)
VERSION_HISTORY to VERSION_RELATIONSHIP (as relationships)	PATH			group <- GROUPS<identification_assignment_relationship, 'relationships'> {CLASS_ID<'version relationship'>}
VERSION_HISTORY to VERSIONABLE_OBJECT (as versions)	PATH			group <- (RELATE_GROUP_2_VO<representation, 'versions'>) (RELATE_GROUP_2_VO<product_definition, 'versions'>) (RELATE_GROUP_2_VO<property_definition, 'versions'>) (RELATE_GROUP_2_VO<shape_aspect, 'versions'>) (RELATE_GROUP_2_VO<product_category, 'versions', >) (RELATE_GROUP_2_VO<document, 'versions'>) (RELATE_GROUP_2_VO<product_definition_relationship, 'versions'>) (RELATE_GROUP_2_VO<shape_aspect_relationship, 'versions'>)
VERSION_RELATIONSHIP	identification_assignment_relationship	41		({ROOT_CLASS<identification_assignment_relationship, 'version relationship'>})
VERSION_RELATIONSHIP to VERSIONABLE_OBJECT (predecessor)	PATH			identification_assignment_relationship identification_assignment_relationship.related_assignment -> (RELATE_ID_2_VO< representation >) (RELATE_ID_2_VO< product_definition >) (RELATE_ID_2_VO< property_definition >) (RELATE_ID_2_VO< shape_aspect >) (RELATE_ID_2_VO< product_category >) (RELATE_ID_2_VO< document >) (RELATE_ID_2_VO< product_definition_relationship >) (RELATE_ID_2_VO< shape_aspect_relationship >)
VERSION_RELATIONSHIP to VERSIONABLE_OBJECT (as successor)	PATH			identification_assignment_relationship identification_assignment_relationship.relating_assignment -> (RELATE_ID_2_VO< representation >)

Application Element	AIM element	Source	Rules	Reference Path
				(RELATE_ID_2_VO< product_definition >) (RELATE_ID_2_VO< property_definition >) (RELATE_ID_2_VO< shape_aspect >) (RELATE_ID_2_VO< product_category >) (RELATE_ID_2_VO< document >) (RELATE_ID_2_VO< product_definition_relationship >) (RELATE_ID_2_VO< shape_aspect_relationship >)
reason	identification_assignment_relationship.description	41		
EVENT #1: VERSIONABLE_OBJECT_CHANGE_EVENT #2: APPROVAL_EVENT #3: CHECK	*see subtypes*			
description	*see subtypes*			
EVENT to DATE_AND_TIME (as caused_when)	PATH			*see subtypes*
EVENT to PERSON_AND_ORGANISATION (as caused_by)	PATH			*see subtypes*
CHANGE	action	41		{ROOT_CLASS<action, ‘change’>}
class	action.description	41		
CHANGE to CHANGE_STATE (as possible_states) #1: CHANGE_STATE = CHANGE_REQUEST #2: CHANGE_STATE = CHANGE_PLAN #3: CHANGE_STATE = CHANGE_REALISATION	PATH			#1: (action action.chosen_method -> action_method <- action_request_solution.method action_request_solution action_request_solution.request -> versioned_action_request) #2: (action action.chosen_method -> action_method <-

Application Element	AIM element	Source	Rules	Reference Path
				action_request_solution.method action_request_solution) #3: (action <- action_relationship.relating_action action_relationship action_relationship.related_action -> action executed_action)
CHANGE_IMPACT	applied_action_request_assignment	216		applied_action_request_assignment <= action_request_assignment {ROOT_CLASS<applied_action_request_assignment, ‘change impact’>}
CHANGE_IMPACT to VERSIONABLE_OBJECT_CHANGE_EVENT (as impact)	PATH			applied_action_request_assignment applied_action_request_assignment.items [i] -> action_request_item action_request_item = action {CLASS_ID<action, ‘versionable object change event’>}
CHANGE_PLAN	action_request_solution	41		{[CLASS<action_request_solution, ‘change plan’, ‘change state’>] [ROOT_CLASS<action_request_solution, ‘change state’>]}
CHANGE_PLAN to CHANGE_IMPACT (as impact)	PATH			action_request_solution action_request_solution.request -> versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment {CLASS_ID<applied_action_request_assignment, ‘change impact’>}
CHANGE_PLAN to CHANGE_REQUEST (as chosen_solution_for)	PATH			action_request_solution action_request_solution.request -> versioned_action_request {CLASS_ID<versioned_action_request, ‘change request’>}
CHANGE_PLAN to CHECK (as checks)	PATH			action_request_solution action_request_solution = action_item action_item <-

Application Element	AIM element	Source	Rules	Reference Path
				applied_action_assignment.items[i] applied_action_assignment action_assignment action_assignment.assigned_action -> action {CLASS_ID<action, ‘check’>}
from CHANGE_STATE				
description	action_request_solution.description	41		
CHANGE_PLAN to CHANGE (as change_reference)	PATH			action_request_solution action_request_solution.method -> action_method <- action.chosen_method action {CLASS_ID<action, ‘change‘>}
CHANGE_PLAN to DATE_AND_TIME (as date_time)	PATH			DAT_TIME_ASSGN<action_request_solution,'date time'>
CHANGE_PLAN to PERSON_AND_ORGANISATION (as author)	PATH			PERS_ORG_ASSGN<action_request_solution, 'author'>
CHANGE_REALIZATION	executed_action	41		{[CLASS<executed_action, ‘change realisation’, ‘change state’>] [ROOT_CLASS<executed_action, ‘change state’>]}
CHANGE_REALIZATION to CHANGE_IMPACT (as impact)	PATH			executed_action action_request_item = executed_action action_request_item <- applied_action_request_assignment.items[i] applied_action_request_assignment {CLASS_ID<applied_action_request_assignment, ‘change impact’>}
CHANGE_REALISATION to CHANGE_PLAN (as realisation_of)	PATH			executed_action <= action action.chosen_method -> action_method <- action_request_solution.method action_request_solution

Application Element	AIM element	Source	Rules	Reference Path
				{CLASS_ID<action_request_solution, ‘change plan’>}
CHANGE_REALIZATION to CHECK (as checks)	PATH			executed_action action_item = executed_action action_item <- applied_action_assignment.items[i] applied_action_assignment action_assignment action_assignment.assigned_action -> action {CLASS_ID<action, ‘check’>}
from CHANGE_STATE				
description	action.description	41		executed_action <= action action.description
CHANGE_REALISATION to CHANGE (as change_reference)	PATH			executed_action <= action <- action_relationship.related_action action_relationship action_relationship.relating_action action {CLASS_ID<action, ‘change‘>}
CHANGE_REALISATION to DATE_AND_TIME (as date_time)	PATH			DAT_TIME_ASSGN<executed_action,'date time'>
CHANGE_REALISATION to PERSON_AND_ORGANISATION (as author)	PATH			PERS_ORG_ASSGN<executed_action, 'author'>
CHANGE_REQUEST	versioned_action_request	41		{[CLASS<versioned_action_request, ‘change request’, ‘change state’>] [ROOT_CLASS<versioned_action_request, ‘change state’>]}
CHANGE_REQUEST to PERSON_AND_ORGANISATION (as initiator)	PATH			PERS_ORG_ASSGN<versioned_action_request, 'initiator'>
CHANGE_REQUEST to	PATH			PERS_ORG_ASSGN<versioned_action_request, 'addressee'>

Application Element	AIM element	Source	Rules	Reference Path
PERSON_AND_ORGANISATION (as addressee)				
problem	versioned_action_request.purpose	41		
solution_description	action_request_solution.description	41		versioned_action_request <- action_request_solution.request action_request_solution action_request_solution.description
CHANGE_REQUEST to CHANGE_IMPACT (as solution_alternatives)	PATH			versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment {CLASS_ID<applied_action_request_assignment, ‘change impact’>}
from CHANGE_STATE				
description	versioned_action_request.description	41		
CHANGE_REQUEST to CHANGE (as change_reference)	PATH			versioned_action_request <- action_request_solution.request action_request_solution action_request_solution.method action_method <- action.chosen_method action {CLASS_ID<action, ‘change‘>}
CHANGE_REQUEST to DATE_AND_TIME (as date_time)	PATH			DAT_TIME_ASSGN< versioned_action_request,'date time'>
CHANGE_REQUEST to PERSON_AND_ORGANISATION (as author)	PATH			PERS_ORG_ASSGN< versioned_action_request, 'author'>
CHANGE_STATE #1: CHANGE_REQUEST #2: CHANGE_PLAN #3: CHANGE_REALIZATION	*see subtypes*			

Application Element	AIM element	Source	Rules	Reference Path
CHANGE_STATE to CHANGE (as change_reference)	PATH			*see subtypes*
description	*see subtypes*			
CHANGE_STATE to DATE_AND_TIME (as date_time)	PATH			*see subtypes*
CHANGE_STATE to PERSON_AND_ORGANISATION (as author)	PATH			*see subtypes*
CHECK	action	41		{[CLASS<action, ‘check’, ‘event’>] [ROOT_CLASS<action, ‘event’>]}
from EVENT				
description	action.description	41		
CHECK to DATE_AND_TIME (as caused_when)	PATH			DAT_TIME_ASSGN<action,'caused when'>
CHECK to PERSON_AND_ORGANISATION (as caused_by)	PATH			PERS_ORG_ASSGN<action, 'caused by'>
VERSIONABLE_OBJECT_CHANGE_EVENT #1: ENVISAGED_VERSION_CREATION #2: VERSION_CREATION #3: VERSION_MODIFICATION #4: VERSION_DELETION	#1, #2, #3, #4: action	41		
from EVENT				
description	action.description	41		
VERSIONABLE_OBJECT_CHANGE_EVENT to DATE_AND_TIME (as caused_when)	PATH			DAT_TIME_ASSGN<action,'caused when'>
VERSIONABLE_OBJECT_CHANGE_EVENT to PERSON_AND_ORGANISATION (as caused_by)	PATH			PERS_ORG_ASSGN<action, 'caused by'>
VERSION_CREATION	action	41		{[CLASS<action, ‘version creation’, ‘versionable_object change event’>]

Application Element	AIM element	Source	Rules	Reference Path
				[CLASS<action, ‘versionable_object change event’, ‘event’>] [ROOT_CLASS<action, ‘event’>]}
VERSION_CREATION to VERSIONABLE_OBJECT (as base)	PATH			action <- action_assignment.assigned_action action_assignment {action_assignment.role = ‘base’} => (RELATE_ACT_2_VO< representation >) (RELATE_ACT_2_VO< product_definition >) (RELATE_ACT_2_VO< property_definition >) (RELATE_ACT_2_VO< shape_aspect >) (RELATE_ACT_2_VO< product_category >) (RELATE_ACT_2_VO< document >) (RELATE_ACT_2_VO< product_definition_relationship >) (RELATE_ACT_2_VO< shape_aspect_relationship >)
VERSION_CREATION to VERSIONABLE_OBJECT (as subject)	PATH			action <- action_assignment.assigned_action action_assignment {action_assignment.role = ‘subject’} => (RELATE_ACT_2_VO< representation >) (RELATE_ACT_2_VO< product_definition >) (RELATE_ACT_2_VO< property_definition >) (RELATE_ACT_2_VO< shape_aspect >) (RELATE_ACT_2_VO< product_category >) (RELATE_ACT_2_VO< document >) (RELATE_ACT_2_VO< product_definition_relationship >) (RELATE_ACT_2_VO< shape_aspect_relationship >)
from EVENT				
description	action.description	41		
VERSION_CREATION to DATE_AND_TIME (as caused_when)	PATH			*see supertype*
VERSION_CREATION to PERSON_AND_ORGANISATION (as caused_by)	PATH			*see supertype*

Application Element	AIM element	Source	Rules	Reference Path
VERSION_DELETION	action	41		{[CLASS<action, ‘version deletion’, ‘versionable_object change event’>] [CLASS<action, ‘versionable_object change event’, ‘event’>] [ROOT_CLASS<action, ‘event’>]}
VERSION_DELETION to VERSIONABLE_OBJECT (as subject)	PATH			action <- action_assignment.assigned_action action_assignment {action_assignment.role = ‘subject’} => (RELATE_ACT_2_VO< representation >) (RELATE_ACT_2_VO< product_definition >) (RELATE_ACT_2_VO< property_definition >) (RELATE_ACT_2_VO< shape_aspect >) (RELATE_ACT_2_VO< product_category >) (RELATE_ACT_2_VO< document >) (RELATE_ACT_2_VO< product_definition_relationship >) (RELATE_ACT_2_VO< shape_aspect_relationship >)
from EVENT				
description	action.description	41		
VERSION_DELETION to DATE_AND_TIME (as caused_when)	PATH			*see supertype*
VERSION_DELETION to PERSON_AND_ORGANISATION (as caused_by)	PATH			*see supertype*
VERSION_MODIFICATION	action	41		{[CLASS<action, ‘version modification’, ‘versionable_object change event’>] [CLASS<action, ‘versionable_object change event’, ‘event’>] [ROOT_CLASS<action, ‘event’>]}
VERSION_MODIFICATION to VERSIONABLE_OBJECT (as base)	PATH			action <- action_assignment.assigned_action action_assignment {action_assignment.role = ‘base’} => (RELATE_ACT_2_VO< representation >) (RELATE_ACT_2_VO< product_definition >)

Application Element	AIM element	Source	Rules	Reference Path
				(RELATE_ACT_2_VO< property_definition >) (RELATE_ACT_2_VO< shape_aspect >) (RELATE_ACT_2_VO< product_category >) (RELATE_ACT_2_VO< document >) (RELATE_ACT_2_VO< product_definition_relationship >) (RELATE_ACT_2_VO< shape_aspect_relationship >)
VERSION_MODIFICATION to VERSIONABLE_OBJECT (as subject)	PATH			action <- action_assignment.assigned_action action_assignment {action_assignment.role = 'subject'} => (RELATE_ACT_2_VO< representation >) (RELATE_ACT_2_VO< product_definition >) (RELATE_ACT_2_VO< property_definition >) (RELATE_ACT_2_VO< shape_aspect >) (RELATE_ACT_2_VO< product_category >) (RELATE_ACT_2_VO< document >) (RELATE_ACT_2_VO< product_definition_relationship >) (RELATE_ACT_2_VO< shape_aspect_relationship >)
from EVENT				
description	action.description	41		
VERSION_MODIFICATION to DATE_AND_TIME (as caused_when)	PATH			*see supertype*
VERSION_MODIFICATION to PERSON_AND_ORGANISATION (as caused_by)	PATH			*see supertype*
ENVISAGED_VERSION_CREATION	action	41		{[CLASS<action, 'envisaged version creation', 'versionable object change event'>] [CLASS<action, 'versionable_object change event', 'event'>] [ROOT_CLASS<action, 'event'>]}
category	action.name	41		
ENVISAGED_VERSION_CREATION to VERSIONABLE_OBJECT (as base)	PATH			action <- action_assignment.assigned_action action_assignment

Application Element	AIM element	Source	Rules	Reference Path
				{action_assignment.role = 'base'} => (RELATE_ACT_2_VO< representation >) (RELATE_ACT_2_VO< product_definition >) (RELATE_ACT_2_VO< property_definition >) (RELATE_ACT_2_VO< shape_aspect >) (RELATE_ACT_2_VO< product_category >) (RELATE_ACT_2_VO< document >) (RELATE_ACT_2_VO< product_definition_relationship >) (RELATE_ACT_2_VO< shape_aspect_relationship >)
from EVENT				
description	action.description	41		
ENVISAGED_VERSION_CREATION to DATE_AND_TIME (as caused_when)	PATH			*see supertype*
ENVISAGED_VERSION_CREATION to PERSON_AND_ORGANISATION (as caused_by)	PATH			*see supertype*

1. prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_prop_def_rep_via_prop_def
2. prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_prop_def
3. prod_def_w_class_id_global_axis_placement_ref_by_exactly_one_prop_def_rep_via_prop_def
4. prod_def_w_class_id_principal_characteristics_ref_by_exactly_one_prop_def_rep_via_prop_def
5. one_or_more_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation
6. prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role_name_eq_IMO_number_or_pennant_hull_number
7. one_rep_item_in_compound_rep_item_if_class_eq_floating_position
8. one_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_property
9. one_rep_item_in_compound_rep_item_if_class_eq_ship_point
10. one_rep_item_in_compound_rep_item_if_class_eq_ship_curve
11. one_rep_item_in_compound_rep_item_if_class_eq_ship_surface
12. one_rep_item_in_compound_rep_item_if_class_eq_offset_point_table_model
13. one_rep_item_in_compound_rep_item_if_class_eq_spacing_position_with_offset
14. one_rep_item_in_compound_rep_item_if_class_eq_spacing_position
15. one_rep_item_in_compound_rep_item_if_class_eq_stability_property
16. one_compound_rep_item_with_class_floating_position_in_compound_rep_item_with_class_hydrostatic_properties_for_constant_floating_position

17. one_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_ship_curve_segment
18. one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_with_class_ship_curve_with_spacing_position
19. one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_with_class_spacing_position_with_offset
20. one_or_more_compound_rep_item_with_class_stability_property_in_compound_rep_item_with_class_stability_properties_for_one_floating_position
21. one_item_in_representation_if_class_eq_hydrostatic_table
22. one_item_in_representation_if_class_eq_midship_tumble
23. one_item_in_representation_if_class_eq_propeller_location
24. one_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model
25. one_item_in_representation_if_class_eq_stability_table
26. prop_def_w_class_id_hydrostatic_definition_ref_by_one_or_more_rep_w_class_id_hydrostatic_table
27. prop_def_w_class_id_stability_definition_ref_by_one_or_more_rep_w_class_id_stability_table
28. one_or_more_items_with_class_hydrostatic_properties_for_constant_floating_position_in_representation_w_class_hydrostatic_table
29. one_or_more_items_with_class_hydrostatic_property_in_representation_w_class_hydrostatic_table
30. one_or_more_items_with_class_in_representation_w_class_offset_point_table_model
31. one_or_more_items_with_class_in_representation_w_class_offset_point_table_model2
32. one_or_more_items_with_class_in_representation_w_class_offset_table_shape_representation
33. one_or_more_items_with_class_in_representation_w_class_stability_table
34. one_or_more_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_properties_for_constant_floating_position
35. one_rep_item_in_compound_rep_item_if_class_eq_stability_properties_for_one_floating_position
36. four_or_more_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model
37. two_or_more_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_knot
38. valid_prod_def_w_classs_moulded_form_in_prod_def_w_class_ship_moulded_form
39. valid_prod_def_rel_w_classs_moulded_form_relationship_in_prod_def_w_class_ship_moulded_form
40. prod_def_rel_w_class_moulded_form_relationship_related_prod_def_shall_have_class_moulded_form
41. prod_def_rel_w_class_moulded_form_relationship_relating_prod_def_shall_have_class_moulded_form
42. dsc_rep_item_w_n_user_def_function_if_dsc_rep_item_w_n_function_and_d_user_defined
43. ID_of_extended_property_definition_with_class_versionable_object_pattern
44. ID_of_product_with_class_definable_object_pattern
45. ID_of_product_definition_with_class_class_and_statutory_designation_pattern
46. ID_of_product_definition_with_class_definable_object_pattern
47. ID_of_product_definition_with_class_global_axis_placement_pattern
48. ID_of_product_definition_with_class_owner_designation_pattern
49. ID_of_product_definition_with_class_principal_characteristics_pattern
50. ID_of_product_definition_with_class_ship_designation_pattern
51. ID_of_product_definition_with_class_shipyard_designation_pattern

52. ID_of_product_definition_relationship_with_class_definable_object_pattern
53. ID_of_product_related_product_category_with_class_ship_type_pattern
54. ID_of_property_definition_with_class_deck_moulded_form_design_parameter_pattern
55. ID_of_property_definition_with_class_hydrostatic_definition_pattern
56. ID_of_property_definition_with_class_local_co_ordinate_system_pattern
57. ID_of_property_definition_with_class_moulded_form_characteristics_definition_pattern
58. ID_of_property_definition_with_class_ship_overall_dimensions_pattern
59. ID_of_property_definition_with_class_spacing_table_pattern
60. ID_of_property_definition_with_class_stability_definition_pattern
61. rep_has_guac
62. max_one_version_id_per_versionable_object
63. id_is_globally_unique
64. prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_class_notation
65. prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_regulation
66. one_item_in_representation_if_name_eq_appendage_moulded_form_design_parameter
67. one_item_in_representation_if_name_eq_bottom_moulded_form_design_parameter
68. one_item_in_representation_if_name_eq_bulb_moulded_form_design_parameter
69. one_item_in_representation_if_name_eq_deck_moulded_form_design_parameter
70. one_item_in_representation_if_name_eq_hull_moulded_form_design_parameter
71. one_item_in_representation_if_name_eq_propeller_moulded_form_design_parameter
72. one_item_in_representation_if_name_eq_rudder_moulded_form_design_parameter
73. one_item_in_representation_if_name_eq_thruster_moulded_form_design_parameter
74. one_item_in_representation_if_name_eq_class_and_statutory_designation
75. one_item_in_representation_if_name_eq_class_notation
76. one_item_in_representation_if_name_eq_class_parameters
77. one_item_in_representation_if_name_eq_global_axis_placement
78. one_item_in_representation_if_name_eq_local_co_ordinate_system_with_position_reference
79. one_item_in_representation_if_name_eq_local_co_ordinate_system
80. one_item_in_representation_if_name_eq_moulded_form_characteristics_definition
81. one_item_in_representation_if_name_eq_moulded_form_function_parameters
82. one_item_in_representation_if_name_eq_principal_characteristics
83. one_item_in_representation_if_name_eq_ship_overall_dimensions
84. zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation
85. zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_hull_moulded_form_design_parameter
86. zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_moulded_form_function_parameters

87. zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_principal_characteristics
88. zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_propeller_moulded_form_design_parameter
89. comp_rep_item_w_class_id_section_of_offset_point_table_ref_by__one_class_ass_w_role_section_identifier
90. prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managing_company
91. prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_ordering_company
92. prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_company
93. prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sign
94. prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_state
95. prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_registration
96. prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_shipyard
97. prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_appendage_moulded_form_design_parameter
98. prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bottom_moulded_form_design_parameter
99. prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bulb_moulded_form_design_parameter
100. prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_deck_moulded_form_design_parameter
101. prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_hull_moulded_form_design_parameter
102. prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_propeller_moulded_form_design_parameter
103. prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_rudder_moulded_form_design_parameter
104. prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_thruster_moulded_form_design_parameter
105. prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_society
106. prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_notation
107. prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_reference
108. prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system
109. prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition
110. prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition
111. prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_transformation_w_name_transformation_to_parent
112. prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_def_rep_w_name_moulded_form_function_parameters
113. prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rel_w_name_thruster_propeller_parameter
114. components_related_to_address

```
(*
```

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and the AIC's and contains the types, entity specializations, rules and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the text for the constructs that are imported from the integrated resources and the AIC's. The definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes that are not imported into the AIM. Requirements stated in the integrated resources that refer to select list items apply exclusively to those items that are imported into the AIM.

```
*)
SCHEMA ship_moulded_form_AIM;

USE FROM approval_schema
    (approval,
    approval_status);

USE FROM action_schema
    (action,
    executed_action,
    action_method,
    action_request_solution,
    action_relationship,
    versioned_action_request);

USE FROM management_resources_schema
    (classification_assignment,
    group_assignment,
    identification_assignment,
    external_identification_assignment,
    organization_assignment,
    document_reference,
    action_assignment,
    action_request_assignment,
    approval_assignment,
    date_and_time_assignment,
    person_and_organization_assignment);

USE FROM aic_non_manifold_surface
    (non_manifold_surface_shape_representation);

USE FROM aic_shell_based_wireframe
    (shell_based_wireframe_shape_representation);

USE FROM document_schema
    (document,
    document_usage_constraint);

USE FROM group_schema
    (group,
    group_relationship);

USE FROM product_definition_schema
    (product,
    product_definition,
    product_definition_relationship,
    product_category,
    product_related_product_category,
    characterized_product_definition);
```

```
USE FROM product_property_definition_schema
    (product_definition_shape,
    property_definition,
    shape_aspect);

USE FROM person_organization_schema
    (organization,
    organizational_project);

USE FROM material_property_definition_schema
    (property_definition_relationship);

USE FROM product_property_representation_schema
    (property_definition_representation,
    shape_representation);

USE FROM geometry_schema
    (axis2_placement_3d,
    axis1_placement,
    cartesian_point,
    elementary_surface,
    placement,
    surface,
    vector,
    point,
    curve,
    plane,
    geometric_representation_item,
    geometric_representation_context);

USE FROM topology_schema
    (vertex,
    vertex_point,
    wire_shell,
    topological_representation_item,
    edge_curve);

USE FROM geometric_model_schema
    (shell_based_wireframe_model,
    face_based_surface_model);

USE FROM external_reference_schema
    (external_source,
    external_source_relationship);

USE FROM qualified_measure_schema
    (descriptive_representation_item);

USE FROM measure_schema
    (named_unit,
    si_unit,
    conversion_based_unit,
    global_unit_assigned_context,
    measure_with_unit,
    length_measure,
    positive_length_measure,
    ratio_measure,
    plane_angle_measure,
    parameter_value,
    area_measure);
```

```
USE FROM representation_schema
    (representation,
    representation_relationship,
    representation_context,
    representation_item,
    value_representation_item,
    compound_representation_item,
    item_defined_transformation,
    list_representation_item);
(*
```

NOTE - The schemas referenced above can be found in the following parts of ISO 10303

action_schema	ISO 10303-41
aic_topologically_bounded_surface	ISO 10303-508
aic_shell_based_wireframe	ISO 10303-502
basic_attribute_schema	ISO 10303-41
external_reference_schema	ISO 10303-41
approval_schema	ISO 10303-41
application_context_schema	ISO 10303-41
document_schema	ISO 10303-41
date_time_schema	ISO 10303-41
geometry_schema	ISO 10303-42
group_schema	ISO 10303-41
management_resources_schema	ISO 10303-41
measure_schema	ISO 10303-41
person_organization_schema	ISO 10303-41
product_definition_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
qualified_measure_schema	ISO 10303-45
representation_schema	ISO 10303-43
topology_schema	ISO 10303-42

5.2.1 Ship moulded form types

5.2.1.1 Ship moulded form type definitions

5.2.1.1.1 classification_item

A classification_item is an aspect of the ship moulded form, to which a class is assigned.

EXPRESS specification

```
*)
TYPE classification_item = SELECT
      (product_definition,
      product_definition_relationship,
      product_definition_shape,
      group,
      extended_property_definition,
      extended_product_definition_shape,
      shape_representation,
      representation_relationship,
      property_definition,
      product,
      extended_product_related_product_category,
      representation,
      compound_representation_item,
      axis2_placement_3d,
```

```
      edge_curve,
      vertex_point,
      measure_with_unit);
END_TYPE;
(*
```

5.2.1.1.2 document_reference_item

A document_reference_item is an aspect of the ship moulded form, to which a document is assigned.

EXPRESS specification

```
*)
TYPE document_reference_item = SELECT
      (property_definition);
END_TYPE;
(*
```

5.2.1.1.3 group_item

A group_item is an aspect of the ship moulded form, to which is assigned to a group.

EXPRESS specification

```
*)
TYPE group_item = SELECT
      (product_definition,
      product_definition_shape,
      product_definition_relationship);
END_TYPE;
(*
```

5.2.1.1.4 organization_item

A organization_item is an aspect of the ship moulded form, to which an organization is assigned.

EXPRESS specification

```
*)
TYPE organization_item = SELECT
      (product_definition,
      property_definition);
END_TYPE;
(*
```

5.2.1.1.5 identification_item

An identification_item is an aspect of the ship moulded form, to which an identifier is assigned.

EXPRESS specification

```
*)
TYPE identification_item = SELECT
      (product_definition,
      extended_property_definition,
      extended_product_definition_shape,
      extended_product_related_product_category,
      compound_representation_item,
      product_definition_relationship,
      group);
END_TYPE;
(*
```

5.2.1.1.6 external_identification_item

An external_identification_item is an aspect of the ship moulded form, which is externally identified.

EXPRESS specification

```
*)
TYPE external_identification_item = SELECT
      (product,
      product_definition);
END_TYPE;
(*
```

5.2.1.1.7 action_request_item

An action_request_item is an aspect of the ship moulded form, which has an action_request assigned.

EXPRESS specification

```
*)
TYPE action_request_item = SELECT
      (action,
      executed_action);
END_TYPE;
(*
```

5.2.1.1.8 action_item

An action_item is an aspect of the ship moulded form, which has an action assigned.

EXPRESS specification

```
*)
TYPE action_item = SELECT
      (action_request_solution,
      executed_action,
      product_definition,
      product_definition_relationship,
      product_definition_shape,
      group,
      property_definition,
      product,
      product_related_product_category);
END_TYPE;
(*
```

5.2.1.1.9 approval_item

An approval_item is an aspect of the ship moulded form, which has an approval assigned.

EXPRESS specification

```
*)
TYPE approval_item = SELECT
    (product_definition,
    product_definition_shape,
    property_definition,
    product_related_product_category);
END_TYPE;
(*
```

5.2.1.1.10 person_and_organization_item

An person_and_organization_item is an aspect of the ship moulded form, which has an person_and_organization assigned.

EXPRESS specification

```
*)
TYPE person_and_organization_item = SELECT
    (action_request_solution,
    executed_action,
    versioned_action_request,
    action);
END_TYPE;
(*
```

5.2.1.1.11 date_and_time_item

A date_and_time_item is an aspect of the ship moulded form, which has a date_and_time assigned.

EXPRESS specification

```
*)
TYPE date_and_time_item = SELECT
    (action_request_solution,
    executed_action,
    versioned_action_request,
    action);
END_TYPE;
(*
```

5.2.2 Ship moulded form entities

5.2.2.1 Ship moulded form entity definitions

5.2.2.1.1 applied_classification_assignment

An applied_classification_assignment assigns a class to a set of classification_items.

EXPRESS specification

```
*)
ENTITY applied_classification_assignment
    SUBTYPE OF (classification_assignment);
    items : SET [1:?] OF classification_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of product_definition, product_definition_relationship, product_definition_shape, group, product_definition_shape, extended_property_definition, extended_product_definition_shape, shape_representation, representation_relationship, property_definition, product, extended_product_related_product_category, representation, compound_representation_item, axis2_placement_3d, edge_curve, vertex_point or measure_with_unit, to which a class is assigned

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the applied_classification_assignment:

— prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_prop_def_rep_via _prop_def (see 5.2.3.1)

— prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.2)

— prod_def_w_class_id_global_axis_placement_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.3)

— prod_def_w_class_id_principal_characteristics_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.4)

— prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role_name_eq_IMO_number_or_pennant_hull_number (see 0)

— one_rep_item_in_compound_rep_item_if_class_eq_floating_position (see 5.2.3.7)

one_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_property (see 5.2.3.8)

one_rep_item_in_compound_rep_item_if_class_eq_offset_point_table_model (see 5.2.3.12)

— one_rep_item_in_compound_rep_item_if_class_eq_spacing_position_with_offset (see 5.2.3.13)

one_rep_item_in_compound_rep_item_if_class_eq_spacing_position (see 5.2.3.14)

— one_rep_item_in_compound_rep_item_if_class_eq_stability_property (see 5.2.3.15)

— one_compound_rep_item_with_class_floating_position_in_compound_rep_item_with_class_hydrostatic_properties_for_constant_floating_position (see 5.2.3.16)

— one_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_ship_curve_segment (see 5.2.3.17)

one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_with_class_spacing_position_with_offset (see 5.2.3.19)

one_or_more_compound_rep_item_with_class_stability_property_in_compound_rep_item_with_class_stability_properties_for_one_floating_position (see 5.2.3.20)

— one_item_in_representation_if_class_eq_hydrostatic_table (see 5.2.3.21)

— one_item_in_representation_if_class_eq_midship_tumble (see 5.2.3.22)

— one_item_in_representation_if_class_eq_propeller_location (see 5.2.3.23)

— one_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model (see 5.2.3.24)

— one_item_in_representation_if_class_eq_stability_table (see 5.2.3.25)

— prop_def_w_class_id_hydrostatic_definition_ref_by_one_or_more_rep_w_class_id_hydrostatic_table (see 5.2.3.26)

— prop_def_w_class_id_stability_definition_ref_by_one_or_more_rep_w_class_id_stability_table (see 5.2.3.27)

— one_or_more_items_with_class_hydrostatic_properties_for_constant_floating_position_in_representation_w_class_hydrostatic_table (see 5.2.3.28)

— one_or_more_items_with_class_hydrostatic_property_in_representation_w_class_hydrostatic_table (see 5.2.3.29)

— one_or_more_items_with_class_in_representation_w_class_offset_point_table_model (see 5.2.3.30)

— one_or_more_items_with_class_in_representation_w_class_offset_point_table_model2 (see 5.2.3.31)

— one_or_more_items_with_class_in_representation_w_class_offset_table_shape_representation (see 5.2.3.32)

— one_or_more_items_with_class_in_representation_w_class_stability_table (see 5.2.3.33)

one_or_more_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_properties_for_constant_floating_position (see 5.2.3.34)

one_rep_item_in_compound_rep_item_if_class_eq_stability_properties_for_one_floating_position (see 5.2.3.35)

— four_or_more_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model see 5.2.3.36)

— two_or_more_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_knot (see 5.2.3.37)

— valid_prod_def_w_classs_moulded_form_in_prod_def_w_class_ship_moulded_form (see 5.2.3.38)

— valid_prod_def_rel_w_classs_moulded_form_relationship_in_prod_def_w_class_ship_moulded_form (see 5.2.3.39)

— prod_def_rel_w_class_moulded_form_relationship_related_prod_def_shall_have_class_moulded_form (see 5.2.3.40)

— prod_def_rel_w_class_moulded_form_relationship_relating_prod_def_shall_have_class_moulded_form (see 5.2.3.41)

— ID_of_extended_property_definition_with_class_versionable_object_pattern (see 5.2.3.43)

— ID_of_product_with_class_definable_object_pattern (see 5.2.3.44)

— ID_of_product_definition_with_class_class_and_statutory_designation_pattern (see 5.2.3.45)

— ID_of_product_definition_with_class_definable_object_pattern (see 5.2.3.46)

— ID_of_product_definition_with_class_global_axis_placement_pattern (see 5.2.3.47)

— ID_of_product_definition_with_class_owner_designation_pattern (see 5.2.3.48)

— ID_of_product_definition_with_class_principal_characteristics_pattern (see 5.2.3.49)

— ID_of_product_definition_with_class_ship_designation_pattern (see 5.2.3.50)

— ID_of_product_definition_with_class_shipyard_designation_pattern (see 5.2.3.51)

— ID_of_product_definition_relationship_with_class_definable_object_pattern (see 5.2.3.52)

— ID_of_product_related_product_category_with_class_ship_type_pattern (see 5.2.3.53)

— ID_of_property_definition_with_class_deck_moulded_form_design_parameter_pattern (see 5.2.3.54)

— ID_of_property_definition_with_class_hydrostatic_definition_pattern (see 5.2.3.55)

— ID_of_property_definition_with_class_local_co_ordinate_system_pattern (see 5.2.3.56)

— ID_of_property_definition_with_class_moulded_form_characteristics_definition_pattern (see 5.2.3.57)

— ID_of_property_definition_with_class_ship_overall_dimensions_pattern see (5.2.3.58)

— ID_of_property_definition_with_class_spacing_table_pattern (see 5.2.3.59)

— ID_of_property_definition_with_class_stability_definition_pattern (see 5.2.3.60)

— id_is_globally_unique (see 5.2.3.63)

— prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_class_notation (see 5.2.3.64)

— prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_regulation (see 5.2.3.65)

— comp_rep_item_w_class_id_section_of_offset_point_table_ref_by__one_class_ass_w_role_section_identifier (see 5.2.3.89)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managing_company (see 5.2.3.90)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_ordering_company (see 5.2.3.91)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_company (see 5.2.3.92)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sign (see 5.2.3.93)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_state (see 5.2.3.94)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_registration (see 5.2.3.95)

— prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_shipyard (see 5.2.3.96)

— prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_appendage_moulded_form_design_parameter (see 5.2.3.97)

— prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bottom_moulded_form_design_parameter (see 5.2.3.98)

— prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bulb_moulded_form_design_parameter (see 5.2.3.99)

— prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_deck_moulded_form_design_parameter (see 5.2.3.100)

— prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_hull_moulded_form_design_parameter (see 5.2.3.101)

— prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_propeller_moulded_form_design_parameter (see 5.2.3.102)

— prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_rudder_moulded_form_design_parameter (see 5.2.3.103)

— prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_thruster_moulded_form_design_parameter (see 5.2.3.104)

— prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_society (see 5.2.3.105)

— prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_notation (see 5.2.3.106)

— prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_reference (see 5.2.3.107)

— prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system (see 5.2.3.108)

— prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition (see 5.2.3.109)

— prop_def_w_class_id_ship_overall_dimensions_ref_by_one_prop_def_rep_w_name_ship_overall_dimensions (see 5.2.3.110)

— prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_transformation_w_name_transformation_to_parent (see 5.2.3.111)

— prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_def_rep_w_name_moulded_form_function_parameters (see 5.2.3.112)

— prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rel_w_name_thruster_propeller_parameter (see 5.2.3.113)

5.2.2.1.2 applied_document_reference

An applied_document_reference provides a reference to a document for the document_reference_item instances.

EXPRESS specification

```
*)
ENTITY applied_document_reference
    SUBTYPE OF (document_reference);
    items : SET [1:?] OF document_reference_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of property_definition, to which a document is assigned

5.2.2.1.3 applied_group_assignment

An applied_group_assignment assigns a group to a set of group_item instances.

EXPRESS specification

```
*)
ENTITY applied_group_assignment
    SUBTYPE OF (group_assignment);
    items : SET [1:?] OF group_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of product_definition, product_definition_shape or product_definition_relationship, which are assigned to a group

5.2.2.1.4 applied_external_identification_assignment

An applied_external_identification_assignment assigns an identifier to a set of externally defined external_identification_item instances.

EXPRESS specification

```
*)
ENTITY applied_external_identification_assignment
    SUBTYPE OF (external_identification_assignment);
    items : SET [1:?] OF external_identification_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of product or product_definition , to which an external_identification is assigned.

5.2.2.1.5 applied_identification_assignment

An applied_identification_assignment assigns an identifier for a set of identification_item instances

EXPRESS specification

```
*)
ENTITY applied_identification_assignment
    SUBTYPE OF (identification_assignment);
    items : SET [1:?] OF identification_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of product_definition, extended_property_definition, extended_product_definition_shape, extended_product_related_product_category, compound_representation_item, product_definition_relationship or group, to which an identifier is assigned

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the applied_ identification_assignment:

— max_one_version_id_per_versionable_object (see 5.2.3.62)

5.2.2.1.6 applied_organization_assignment

An applied_organization_assignment assigns an organization to a set of organization_item instances.

EXPRESS specification

```
*)
ENTITY applied_organization_assignment
    SUBTYPE OF (organization_assignment);
    items : SET [1:?] OF organization_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of product_definition or property_definition, to which an organization is assigned

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to theapplied_organization_assignment:

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managing_company (see 5.2.3.90)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_ordering_company (see 5.2.3.91)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_company (see 5.2.3.92)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sign (see 5.2.3.93)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_state (see 5.2.3.94)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_registration (see 5.2.3.95)

— prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_shipyard (see 5.2.3.96)

— prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_society (see 5.2.3.105)

5.2.2.1.7 applied_action_request_assignment

An applied_action_request_assignment assigns an action_request for a set of action_request_item instances

EXPRESS specification

```
*)
ENTITY applied_action_request_assignment
    SUBTYPE OF (action_request_assignment);
```

```
    items : SET [1:?] OF action_request_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of action or executed_action, to which an action_request is assigned

5.2.2.1.8 applied_action_assignment

An applied_action_assignment assigns an action for a set of action_item instances

EXPRESS specification

```
*)
ENTITY applied_action_assignment
    SUBTYPE OF (action_assignment);
    items : SET [1:?] OF action_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of action_request_solution, executed_action, product_definition, product_definition_relationship, product_definition_shape, group,property_definition, product, product_related_product_category, to which an action is assigned

5.2.2.1.9 applied_approval_assignment

An applied_approval_assignment assigns an approval for a set of approval_item instances

EXPRESS specification

```
*)
ENTITY applied_approval_assignment
    SUBTYPE OF (approval_assignment);
    items : SET [1:?] OF approval_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of product_definition, product_definition_shape, property_definition, product_related_product_category, to which an approval is assigned.

5.2.2.1.10 applied_person_and_organization_assignment

An applied_person_and_organization_assignment assigns a person and an organization to a set of person_and_organization_item instances.

EXPRESS specification

```
*)
ENTITY applied_person_and_organization_assignment
    SUBTYPE OF (person_and_organization_assignment);
    items : SET [1:?] OF person_and_organization_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of action_request_solution, executed_action, versioned_action_request, action, to which an person_and_organization is assigned

5.2.2.1.11 applied_date_and_time_assignment

An applied_date_and_time_assignment assigns a date and time to a set of date_and_time_item instances.

EXPRESS specification

```
*)
ENTITY applied_date_and_time_assignment
    SUBTYPE OF (date_and_time_assignment);
    items : SET [1:?] OF date_and_time_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the instances of action_request_solution, executed_action, versioned_action_request, action, to which a date_and_time is assigned

5.2.2.1.12 class

A class is a group that is used to classify any aspect of the ship moulded form.

EXPRESS specification

```
*)
ENTITY class
    SUBTYPE OF (group);
END_ENTITY;
(*
```

5.2.2.1.13 extended_property_definition

A extended_property_definition is a property_definition with a derived id attribute.

EXPRESS specification

```
*)
ENTITY extended_property_definition
    SUBTYPE OF (property_definition);
DERIVE
    id: identifier := get_extended_property_definition_id_value(SELF);
END_ENTITY;
(*
```

Attribute definitions:

id: the derived id attribute provided by an applied_identification_assignment

5.2.2.1.14 extended_product_definition_shape

A extended_product_definition_shape is a product_definition_shape with a derived id attribute.

EXPRESS specification

```
*)
ENTITY extended_product_definition_shape
    SUBTYPE OF (product_definition_shape);
DERIVE
    id: identifier :=
get_extended_product_definition_shape_id_value(SELF);
END_ENTITY;
(*
```

Attribute definitions:

id: the derived id attribute provided by an applied_identification_assignment

Formal propositions:

5.2.2.1.15 extended_product_related_product_category

A extended_product_related_product_category is a product_related_product_category with a derived id attribute.

EXPRESS specification

```
*)
ENTITY extended_product_related_product_category
    SUBTYPE OF (product_related_product_category);
DERIVE
    id: identifier :=
get_extended_product_related_product_category_id_value(SELF);
END_ENTITY;
(*
```

Attribute definitions:

id: the derived id attribute provided by an applied_identification_assignment

5.2.2.2 Ship moulded form imported entity modifications

1.1.1.1.1 product_definition

The base definition of the product_definition is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the product_definition entity:

— prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.1)

— prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.2)

— prod_def_w_class_id_global_axis_placement_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.3)

— prod_def_w_class_id_principal_characteristics_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.4)

— prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role_name_eq_IMO_number_or_pennant_hull_number (see 0)

— valid_prod_def_w_classs_moulded_form_in_prod_def_w_class_ship_moulded_form (see 5.2.3.38)

— valid_prod_def_rel_w_classs_moulded_form_relationship_in_prod_def_w_class_ship_moulded_form (see 5.2.3.39)

— prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_class_notation (see 5.2.3.64)

— prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_regulation (see 5.2.3.65)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managing_company (see 5.2.3.90)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_ordering_company (see 5.2.3.91)

— prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_company (see 5.2.3.92)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sign (see 5.2.3.93)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_state (see 5.2.3.94)

— prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_registration (see 5.2.3.95)

— prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_shipyard (see 5.2.3.96)

1.1.1.1.2 property_definition

The base definition of the property_definition is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the property_definition entity:

— prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.1)

— prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.2)

— prod_def_w_class_id_global_axis_placement_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.3)

— prod_def_w_class_id_principal_characteristics_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.4)

— prop_def_w_class_id_hydrostatic_definition_ref_by_one_or_more_rep_w_class_id_hydrostatic_table (see 5.2.3.26)

— prop_def_w_class_id_stability_definition_ref_by_one_or_more_rep_w_class_id_stability_table (see 5.2.3.27)

— prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_class_notation (see 5.2.3.64)

— prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_regulation (see 5.2.3.65)

— prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_appendage_moulded_form_design_parameter (see 5.2.3.97)

— prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bottom_moulded_form_design_parameter (see 5.2.3.98)

— prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bulb_moulded_form_design_parameter (see 5.2.3.99)

— prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_deck_moulded_form_design_parameter (see 5.2.3.100)

— prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_hull_moulded_form_design_parameter (see 5.2.3.101)

— prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_propeller_moulded_form_design_parameter (see 5.2.3.102)

— prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_rudder_moulded_form_design_parameter (see 5.2.3.103)

— prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_thruster_moulded_form_design_parameter (see 5.2.3.104)

— prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_society (see 5.2.3.105)

— prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_notation (see 5.2.3.106)

— prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_reference (see 5.2.3.107)

— prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system (see 5.2.3.108)

— prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition (see 5.2.3.109)

— prop_def_w_class_id_ship_overall_dimensions_ref_by_one_prop_def_rep_w_name_ship_overall_dimensions (see 5.2.3.110)

— prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_transformation_w_name_transformation_to_parent (see 5.2.3.111)

— prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_def_rep_w_name_moulded_form_function_parameters (see 5.2.3.112)

— prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rel_w_name_thruster_propeller_parameter (see 5.2.3.113)

1.1.1.1.3 property_definition_representation

The base definition of the property_definition_representation is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the property_definition_representation entity:

— prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.1)

— prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.2)

— prod_def_w_class_id_global_axis_placement_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.3)

— prod_def_w_class_id_principal_characteristics_ref_by_exactly_one_prop_def_rep_via_prop_def (see 5.2.3.4)

— prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_appendage_moulded_form_design_parameter (see 5.2.3.97)

— prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bottom_moulded_form_design_parameter (see 5.2.3.98)

— prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bulb_moulded_form_design_parameter (see 5.2.3.99)

— prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_deck_moulded_form_design_parameter (see 5.2.3.100)

— prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_hull_moulded_form_design_parameter (see 5.2.3.101)

— prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_propeller_moulded_form_design_parameter (see 5.2.3.102)

— prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_rudder_moulded_form_design_parameter (see 5.2.3.103)

— prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_thruster_moulded_form_design_parameter (see 5.2.3.104)

— prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_notation (see 5.2.3.106)

— prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_reference (see 5.2.3.107)

— prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system (see 5.2.3.108)

— prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition (see 5.2.3.109)

— prop_def_w_class_id_ship_overall_dimensions_ref_by_one_prop_def_rep_w_name_ship_overall_dimensions (see 5.2.3.110)

— prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_def_rep_w_name_moulded_form_function_parameters (see 5.2.3.112)

— prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rel_w_name_thruster_propeller_parameter (see 5.2.3.113)

1.1.1.1.4 representation

The base definition of the representation is given in ISO 10303-43. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the representation:

— one_or_more_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation (see 5.2.3.5)

— one_item_in_representation_if_class_eq_hydrostatic_table (see 5.2.3.21)

— one_item_in_representation_if_class_eq_midship_tumble (see 5.2.3.22)

— one_item_in_representation_if_class_eq_propeller_location (see 5.2.3.23)

— one_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model (see 5.2.3.24)

— one_item_in_representation_if_class_eq_stability_table (see 5.2.3.25)

— one_or_more_items_with_class_hydrostatic_properties_for_constant_floating_position_in_representation_w_class_hydrostatic_table (see 5.2.3.28)

— one_or_more_items_with_class_hydrostatic_property_in_representation_w_class_hydrostatic_table (see 5.2.3.29)

— one_or_more_items_with_class_in_representation_w_class_offset_point_table_model (see 5.2.3.30)

— one_or_more_items_with_class_in_representation_w_class_offset_point_table_model2 (see 5.2.3.31)

— one_or_more_items_with_class_in_representation_w_class_offset_table_shape_representation (see 5.2.3.32)

— one_or_more_items_with_class_in_representation_w_class_stability_table (see 5.2.3.33)

— four_or_more_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model see 5.2.3.36)

— dsc_rep_item_w_n_user_def_function_if_dsc_rep_item_w_n_function_and_d_user_defined (see 5.2.3.42)

— rep_has_guac (see 5.2.3.61)

— one_item_in_representation_if_name_eq_appendage_moulded_form_design_parameter (see 5.2.3.66)

— one_item_in_representation_if_name_eq_bottom_moulded_form_design_parameter (see 5.2.3.67)

— one_item_in_representation_if_name_eq_bulb_moulded_form_design_parameter (see 5.2.3.68)

— one_item_in_representation_if_name_eq_deck_moulded_form_design_parameter (see 5.2.3.69)

— one_item_in_representation_if_name_eq_hull_moulded_form_design_parameter (see 5.2.3.70)

— one_item_in_representation_if_name_eq_propeller_moulded_form_design_parameter (see 5.2.3.71)

— one_item_in_representation_if_name_eq_rudder_moulded_form_design_parameter (see 5.2.3.72)

— one_item_in_representation_if_name_eq_thruster_moulded_form_design_parameter (see 5.2.3.73)

— one_item_in_representation_if_name_eq_class_and_statutory_designation (see 5.2.3.74)

— one_item_in_representation_if_name_eq_class_notation (see 5.2.3.75)

— one_item_in_representation_if_name_eq_class_parameters (see 5.2.3.76)

— one_item_in_representation_if_name_eq_global_axis_placement (see 5.2.3.77)

— one_item_in_representation_if_name_eq_local_co_ordinate_system_with_position_reference (see 5.2.3.78)

— one_item_in_representation_if_name_eq_local_co_ordinate_system (see 5.2.3.79)

— one_item_in_representation_if_name_eq_moulded_form_characteristics_definition (see 5.2.3.80)

— one_item_in_representation_if_name_eq_moulded_form_function_parameters (see 5.2.3.81)

— one_item_in_representation_if_name_eq_principal_characteristics (see 5.2.3.82)

— one_item_in_representation_if_name_eq_ship_overall_dimensions (see 5.2.3.83)

— zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation (see 5.2.3.84)

— zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_hull_moulded_form_design_parameter (see 5.2.3.85)

— zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_moulded_form_function_parameters (see 5.2.3.86)

— zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_principal_characteristics (see 5.2.3.87)

— zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_propeller_moulded_form_design_parameter (see 5.2.3.88)

1.1.1.1.5 compound_representation_item

The base definition of the compound_representation_item is given in ISO 10303-43. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the compound_representation_item:

— one_rep_item_in_compound_rep_item_if_class_eq_floating_position (see 5.2.3.7)

one_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_property (see 5.2.3.8)

one_rep_item_in_compound_rep_item_if_class_eq_offset_point_table_model (see 5.2.3.12)

one_rep_item_in_compound_rep_item_if_class_eq_spacing_position_with_offset (see 5.2.3.13)

one_rep_item_in_compound_rep_item_if_class_eq_spacing_position (see 5.2.3.14)

one_rep_item_in_compound_rep_item_if_class_eq_stability_property (see 5.2.3.15)

one_compound_rep_item_with_class_floating_position_in_compound_rep_item_with_class_hydrostatic_properties_for_constant_floating_position (see 5.2.3.16)

one_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_ship_curve_segment (see 5.2.3.17)

one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_with_class_spacing_position_with_offset (see 5.2.3.19)

one_or_more_compound_rep_item_with_class_stability_property_in_compound_rep_item_with_class_stability_properties_for_one_floating_position (see 5.2.3.20)

one_or_more_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_properties_for_constant_floating_position (see 5.2.3.34)

one_rep_item_in_compound_rep_item_if_class_eq_stability_properties_for_one_floating_position (see 5.2.3.35)

two_or_more_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_knot (see 5.2.3.37)

comp_rep_item_w_class_id_section_of_offset_point_table_ref_by__one_class_ass_w_role_section_identifier (see 5.2.3.89)

1.1.1.1.6 product_definition_relationship

The base definition of the product_definition_relationship is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the product_definition_relationship

— prod_def_rel_w_class_moulded_form_relationship_related_prod_def_shall_have_class_moulded_form (see 5.2.3.40)

— prod_def_rel_w_class_moulded_form_relationship_relating_prod_def_shall_have_class_moulded_form (see 5.2.3.41)

1.1.1.1.7 item_defined_transformation

The base definition of the item_defined_transformation is given in ISO 10303-43. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the item_defined_transformation.

— prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_transformation_w_name_transformation_to_parent (see 5.2.3.111)

1.1.1.1.8 external_source

The base definition of the external_source is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to theexternal_source:

— components_related_to_address (see 5.2.3.114)

5.2.3 Ship moulded rule definitions

5.2.3.1 prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_prop_def_rep_via_prop_def

Each instance of product_definition with a class id ‘class and statutory designation’ should be referenced by one property_definition_representation with the name ‘class and statutory designation’ via a property_definition.

```
*)
RULE
prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_pr
op_def_rep_via_prop_def FOR (product_definition, property_definition,
property_definition_representation, applied_classification_assignment);

LOCAL
    c_a_set: SET OF applied_classification_assignment := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  SET OF property_definition := [];
    t4_set:  SET OF product_definition := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’class and
statutory designation’*)
```

```
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
    i.ASSIGNED_CLASSIFICATION.NAME = 'class and statutory designation');

(*get all instances of product_definition that have class id 'class and
statutory designation'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all property_definition_representation instances with name 'class
and statutory designation'*)

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
          i.NAME = 'class and statutory designation');

(*get all property_definition instances which are the .definition of the
property_definition_representation*)

REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

(*get all product_definition instances which are the .definition of the
property_definition*)

REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

(*compare both lists with product_definition instances which have to be
identical*)

violation := t1_set = t4_set;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that each instance of product_definition with a class id 'class and statutory designation' is referenced by one property_definition_representation with the name 'class and statutory designation' via a property_definition.

5.2.3.2 prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_prop_def

Each instance of product_definition with a class id ‘class parameters’ should be referenced by one property_definition_representation with the name ‘class parameters’ via a property_definition.

```
*)
RULE
prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_
prop_def FOR (product_definition, property_definition,
property_definition_representation, applied_classification_assignment);

LOCAL
    c_a_set: SET OF applied_classification_assignment := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  SET OF property_definition := [];
    t4_set:  SET OF product_definition := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’class
parameters’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
    i.ASSIGNED_CLASSIFICATION.NAME = ’class parameters’);

(*get all instances of product_definition that have class id ’class
parameters’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all property_definition_representation instances with name ’class
parameters’*)

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
          i.NAME = ’class parameters’);

(*get all property_definition instances which are the .definition of the
property_definition_representation*)

REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

(*get all product_definition instances which are the .definition of the
property_definition*)

REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

(*compare both lists with product_definition instances which have to be
identical*)

violation := t1_set = t4_set;

WHERE
```

```
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that each instance of product_definition with a class id ‘class parameters’ is referenced by one property_definition_representation with the name ‘class parameters’ via a property_definition.

5.2.3.3 prod_def_w_class_id_global_axis_placement_ref_by_exactly_one_prop_def_rep_via_prop_def

Each instance of product_definition with a class id ‘global axis placement’ should be referenced by one property_definition_representation with the name ‘global axis placement’ via a property_definition.

```
*)
RULE
prod_def_w_class_id_global_axis_placement_designation_ref_by_exactly_one_
prop_def_rep_via_prop_def FOR (product_definition, property_definition,
property_definition_representation, applied_classification_assignment);

LOCAL
    c_a_set: SET OF applied_classification_assignment := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  SET OF property_definition := [];
    t4_set:  SET OF product_definition := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’global axis
placement’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
    i.ASSIGNED_CLASSIFICATION.NAME = ’global axis placement’);

(*get all instances of product_definition that have class id ’global axis
placement’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all property_definition_representation instances with name ’global
axis placement’*)

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
          i.NAME = ’global axis placement’);
```

```
(*get all property_definition instances which are the .definition of the
property_definition_representation*)

REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

(*get all product_definition instances which are the .definition of the
property_definition*)

REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

(*compare both lists with product_definition instances which have to be
identical*)

violation := t1_set = t4_set;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that each instance of product_definition with a class id ‘global axis placement’ is referenced by one property_definition_representation with the name ‘global axis placement’ via a property_definition.

5.2.3.4 prod_def_w_class_id_principal_characteristics_ref_by_exactly_one_prop_def_rep_via_prop_def

Each instance of product_definition with a class id ‘principal characteristics’ should be referenced by one property_definition_representation with the name ‘principal characteristics’ via a property_definition.

```
*)
RULE
prod_def_w_class_id_principal_characteristics_designation_ref_by_exactly_
one_prop_def_rep_via_prop_def FOR (product_definition,
property_definition, property_definition_representation,
applied_classification_assignment);

LOCAL
    c_a_set: SET OF applied_classification_assignment := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  SET OF property_definition := [];
    t4_set:  SET OF product_definition := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;
```

```
(*get all classification_assignment instances with id 'principal
characteristics'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
    i.ASSIGNED_CLASSIFICATION.NAME = 'principal characteristics');

(*get all instances of product_definition that have class id 'principal
characteristics'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all property_definition_representation instances with name
'principal characteristics'*)

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
          i.NAME = 'global axis placement');

(*get all property_definition instances which are the .definition of the
property_definition_representation*)

REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

(*get all product_definition instances which are the .definition of the
property_definition*)

REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

(*compare both lists with product_definition instances which have to be
identical*)

violation := t1_set = t4_set;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that each instance of product_definition with a class id 'principal characteristics' is referenced by one property_definition_representation with the name'principal characteristics' via a property_definition.

5.2.3.5 one_or_more_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation

Forces the .items attribute of a representation to have one or more representation_items whose .name attribute has the value ‘class notations hull’ and ‘class notations machinery’ if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘class notation’.

```
*)
RULE
one_or_more_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation
FOR (representation);

LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class notations hull', 'class
notations machinery'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation which are used by a
property_definition_representation with name equal to 'class notation'*)

reps := QUERY(temp_rep <* representation |
    SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRES
ENTATION')
    | (temp_prop_def_rep.name = 'class notation'))) > 0 );

(*iterate over all representations found above; stop, if one of them has
not exactly one rep_item for each name in the arg_list *)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name = arg_list[j])) < 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that the .items attribute of a representation has one or more representation_items whose .name attribute has the value ‘class notations hull’ and ‘class notations machinery’ if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘class notation’. prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role_name_eq_IMO_number_or_pennant_hull_number

5.2.3.6 prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role_name_eq_IMO_number_or_pennant_hull_number

Ensures that an instance of product_definition with class id ‘ship designation’ should be referenced by one applied_classification_assignment instance via the .items attribute, where the attribute of applied_classification_assignment .role.name has the value ‘IMO number’ or ‘pennant hull number’.

```
*)
RULE
prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role
_name_eq_IMO_number_or_pennant_hull_number FOR(product_definition,
applied_classification_assignment);

LOCAL
    c_a_set: SET OF applied_classification_assignment := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_classification_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship
designation’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship designation’);

(*get all instances of T1 that have class id ’ship designation’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set: get the
applied_classification_assignment instances that are referencing a
product_definition instance via .ITEMS, filter out those
applied_classification_assignment instances whose attribute .ROLE.NAME
has the value ’IMO number’ or ’pennant hull number’, check if their
number is equal 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’APPLIED_IDENTIFICATION_ASSIGNMENT’ + ’.’ + ’ITEMS’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | (i.ROLE.NAME = ’IMO
number’)OR (i.ROLE.NAME = ’pennant hull number’) )) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘ship designation’ is referenced by one applied_classification_assignment instance via the .items attribute, where the attribute of applied_classification_assignment .role.name has the value ‘IMO number’ or ‘pennant hull number’.

5.2.3.7 one_rep_item_in_compound_rep_item_if_class_eq_floating_position

Ensures that an instance of compound_representation_item with class id ‘floating position’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_floating_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’moulded form displacement’, ’draught
at amidships’, ’length of waterline’, ’breadth of waterline’, ’angle of
trim’, ’angle of heel’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’floating
position’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’floating position’);

(* get all instances of compound_representation_item that have class id
’floating position’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘floating position’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘moulded form displacement‘, ‘draught at amidships‘, ‘length of waterline‘, ‘breadth of waterline‘, ‘angle of trim‘, ‘angle of heel‘].

5.2.3.8 one_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_property

Ensures that an instance of compound_representation_item with class id ‘hydrostatic property’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_property
FOR (compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’property type’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’hydrostatic
property’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’hydrostatic property’);

(*get all instances of compound_representation_item that have class id
’hydrostatic property’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘hydrostatic property’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘property type‘].

5.2.3.9 one_rep_item_in_compound_rep_item_if_class_eq_ship_point

Ensures that an instance of compound_representation_item with class id ‘ship point’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_ship_point FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’point shape’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship point’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship point’);

(*get all instances of compound_representation_item that have class id
’ship point’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘ship point’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘point shape‘].

5.2.3.10 one_rep_item_in_compound_rep_item_if_class_eq_ship_curve

Ensures that an instance of compound_representation_item with class id ‘ship curve’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_ship_curve FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['side condition', 'curve shape'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'ship curve'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve');

(*get all instances of compound_representation_item that have class id
'ship curve'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘ship curve’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘side condition‘, ‘curve shape‘].

5.2.3.11 one_rep_item_in_compound_rep_item_if_class_eq_ship_surface

Ensures that an instance of compound_representation_item with class id ‘ship surface’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_ship_surface FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’surface shape’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship surface’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship surface’);

(*get all instances of compound_representation_item that have class id
’ship surface’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘ship surface’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘surface shape‘].

5.2.3.12 one_rep_item_in_compound_rep_item_if_class_eq_offset_point_table_model

Ensures that an instance of compound_representation_item with class id ‘offset point table model’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_rep_item_in_compound_rep_item_if_class_eq_offset_point_table_model
FOR (compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’offset point table type’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’offset point table
model’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’offset point table model’);

(*get all instances of compound_representation_item that have class id
’offset point table model’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘offset point table model’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘offset point table type‘].

5.2.3.13 one_rep_item_in_compound_rep_item_if_class_eq_spacing_position_with_offset

Ensures that an instance of compound_representation_item with class id ‘spacing position with offset’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_rep_item_in_compound_rep_item_if_class_eq_spacing_position_with_offse
t FOR (compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’offset’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’spacing position
with offset’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’spacing position with
      offset’);

(*get all instances of compound_representation_item that have class id
’spacing position with offset’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘spacing position with offset’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘offset‘].

5.2.3.14 one_rep_item_in_compound_rep_item_if_class_eq_spacing_position

Ensures that an instance of compound_representation_item with class id ‘spacing position’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_spacing_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := [’position number’, ’position’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’spacing
position’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’spacing position’);

(*get all instances of compound_representation_item that have class id
’spacing position’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘spacing position’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘position number‘, ‘position‘].

5.2.3.15 one_rep_item_in_compound_rep_item_if_class_eq_stability_property

Ensures that an instance of compound_representation_item with class id ‘stability property’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_rep_item_in_compound_rep_item_if_class_eq_stability_property FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['angle of heel', 'righting arm',
'centre of buoyancy'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'stability
property'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability property');

(*get all instances of compound_representation_item that have class id
'stability property'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘stability property’ has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘angle of heel‘, ‘righting arm‘, ‘centre of buoyancy‘].

5.2.3.16 one_compound_rep_item_with_class_floating_position_in_compound_rep_item_with_class_hydrostatic_properties_for_constant_floating_position

Ensures that an instance of compound_representation_item with class id ‘hydrostatic properties for constant floating position’ has in its attribute .item_element in the list_representation_item one compound_representation_item with class id ‘floating position’.

```
*)
RULE
one_compound_rep_item_with_class_floating_position_in_compound_rep_item_w
ith_class_hydrostatic_properties_for_constant_floating_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’hydrostatic
properties for constant floating position’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’hydrostatic properties for
      constant floating position’);

(*get all instances of compound_representation_item that have class id
’hydrostatic properties for constant floating position’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’floating
position’*)

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’floating position’);

(* get all instances of compound_representation_item that have class id
’floating position’*)

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;
```

```
(*iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second
list of compound_representation_item is equal 1*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘hydrostatic properties for constant floating position’ has in its attribute .item_element in the list_representation_item one compound_representation_item with class id ‘floating position’.

5.2.3.17 one_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_ship_curve_segment

Ensures that an instance of compound_representation_item with class id ‘ship curve segment‘ has in its attribute .item_element in the list_representation_item one compound_representation_item with class id ‘ship curve‘.

```
*)
RULE
one_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_cla
ss_ship_curve_segment FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship curve
segment’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship curve segment’);

(*get all instances of compound_representation_item that have class id
’ship curve segment’*)
```

```
REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'ship curve'*)

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve');

(* get all instances of compound_representation_item that have class id
'ship curve'*)

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;

(*iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second
list of compound_representation_item is equal 1*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id 'ship curve segment' has in its attribute .item_element in the list_representation_item one compound_representation_item with class id 'ship curve'.

5.2.3.18 one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_with_class_ship_curve_with_spacing_position

Ensures that an instance of compound_representation_item with class id 'ship curve with spacing position' has in its attribute .item_element in the list_representation_item one compound_representation_item with class id 'spacing position'.

```
*)
RULE
one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_wi
th_class_ship_curve_with_spacing_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'ship curve with
spacing position'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve with spacing
      position');

(*get all instances of compound_representation_item that have class id
'ship curve with spacing position'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'spacing
position'*)

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'spacing position');

(* get all instances of compound_representation_item that have class id
'spacing position'*)

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;

(*iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second
list of compound_representation_item is equal 1*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘ship curve with spacing position‘ has in its attribute .item_element in the list_representation_item one compound_representation_item with class id ‘spacing position‘.

5.2.3.19 one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_with_class_spacing_position_with_offset

Ensures that an instance of compound_representation_item with class id ‘spacing position with offset’ has in its attribute .item_element in the list_representation_item one compound_representation_item with class id ‘spacing position’.

```
*)
RULE
one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_wi
th_class_spacing_position_with_offset FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’spacing position
with offset’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’spacing position with
      offset’);

(*get all instances of compound_representation_item that have class id
’spacing position with offset’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’spacing
position’*)

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’spacing position’);

(* get all instances of compound_representation_item that have class id
’spacing position’*)

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;
```

```
(*iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second
list of compound_representation_item is equal 1*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘spacing position with offset’ has in its attribute .item_element in the list_representation_item one compound_representation_item with class id ‘spacing position’.

5.2.3.20 one_or_more_compound_rep_item_with_class_stability_property_in_compound_rep_item_with_class_stability_properties_for_one_floating_position

Ensures that an instance of compound_representation_item with class id ‘stability properties for one floating position’ has in its attribute .item_element in the list_representation_item one or more compound_representation_item with class id ‘stability property’.

```
*)
RULE
one_or_more_compound_rep_item_with_class_stability_property_in_compound_r
ep_item_with_class_stability_properties_for_one_floating_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’stability
properties for one floating position’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’stability properties for one
      floating position’);

(*get all instances of compound_representation_item that have class id
’stability properties for one floating position’*)
```

```
REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'stability
property'*)

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability property');

(* get all instances of compound_representation_item that have class id
'stability property'*)

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;

(*iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second
list of compound_representation_item is equal 1*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id 'stability properties for one floating position' has in its attribute .item_element in the list_representation_item one or more compound_representation_item with class id 'stability property'.

5.2.3.21 one_item_in_representation_if_class_eq_hydrostatic_table

Ensures that an instance of representation with class id 'hydrostatic table' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_class_eq_hydrostatic_table FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['mean shell thickness'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'hydrostatic
table'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

(*get all instances of representation that have class id 'hydrostatic
table'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
     END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'hydrostatic table' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['mean shell thickness'].

5.2.3.22 one_item_in_representation_if_class_eq_midship_tumble

Ensures that an instance of representation with class id 'midship tumble' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_class_eq_midship_tumble FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF REPRESENTATION := [];
```

```
    arg_list:  LIST OF STRING := ['tumble out at bottom', 'tumble in at
top', 'tumble out at side', 'tumble in at side'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'midship tumble'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'midship tumble');

(*get all instances of representation that have class id 'midship
tumble'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
     END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id ‘midship tumble’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘tumble out at bottom‘, ‘tumble in at top‘, ‘tumble out at side‘, ‘tumble in at side‘].

5.2.3.23 one_item_in_representation_if_class_eq_propeller_location

Ensures that an instance of representation with class id ‘propeller location’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_class_eq_propeller_location FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['shaft line inclination x', 'shaft line
inclination y', 'shaft line location', 'propeller location'];
```

```
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'propeller
location'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'propeller location');

(*get all instances of representation that have class id 'propeller
location'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
     END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'propeller location' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['shaft line inclination x', 'shaft line inclination y', 'shaft line location', 'propeller location'].

5.2.3.24 one_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model

Ensures that an instance of representation with class id 'ship moulded form wireframe model' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model
FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['wireframe'];
```

```
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'ship moulded form
wireframe model'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form wireframe
      model');

(*get all instances of representation that have class id 'ship moulded
form wireframe model'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
     END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'ship moulded form wireframe model' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['wireframe'].

5.2.3.25 one_item_in_representation_if_class_eq_stability_table

Ensures that an instance of representation with class id 'stability table' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_class_eq_stability_table FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['mean shell thickness'];
    violation: BOOLEAN := FALSE;
END_LOCAL;
```

```
(*get all classification_assignment instances with id 'stability table'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability table');

(*get all instances of representation that have class id 'stability
table'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
     END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'stability table' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['mean shell thickness'].

5.2.3.26 prop_def_w_class_id_hydrostatic_definition_ref_by_one_or_more_rep_w_class_id_hydrostatic_table

Ensures that an instance of property_definition with class id 'hydrostatic definition' is referenced by one representation with a class id 'hydrostatic table' via a property_definition_representation.

```
*)
RULE
prop_def_w_class_id_hydrostatic_definition_ref_by_one_or_more_rep_w_class
_id_hydrostatic_table FOR (property_definition,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF representation := [];
    t3_set:  SET OF property_definition_representation := [];
    t4_set:  SET OF property_definition := [];
    violation: BOOLEAN := FALSE;
```

```
END_LOCAL;

(*get all classification_assignment instances with id 'hydrostatic
definition'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic definition');

(*get all instances of property_definition that have class id
'hydrostatic definition'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'hydrostatic
table'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

(*get all instances of representation that have class id 'hydrostatic
table'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all property_definition_representation instances which have as the
.used_representation the representation instances that have class id
'hydrostatic table'*)

REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + USEDIN(t2_set[i],
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'
);
END_REPEAT;

(*get all property_definition instances which are the .definition of the
property_definition *)

REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

(*compare both lists with property_definition instances which have to be
identical*)

violation := t1_set = t4_set;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

property_definition: the set of all instances of property_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'hydrostatic definition' is referenced by one representation with a class id 'hydrostatic table' via a property_definition_representation.

5.2.3.27 prop_def_w_class_id_stability_definition_ref_by_one_or_more_rep_w_class_id_stability_table

Ensures that an instance of property_definition with class id 'stability definition' is referenced by one representation with a class id 'stability table' via a property_definition_representation.

```
*)
RULE
prop_def_w_class_id_stability_definition_ref_by_one_or_more_rep_w_class_i
d_stability_table FOR (property_definition,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF representation := [];
    t3_set:  SET OF property_definition_representation := [];
    t4_set:  SET OF property_definition := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'stability
definition'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability definition');

(*get all instances of property_definition that have class id 'stability
definition'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'stability table'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability table');

(*get all instances of representation that have class id 'stability
table'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;
```

```
(*get all property_definition_representation instances which have as the
.used_representation the representation instances that have class id
'stability table'*)

REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + USEDIN(t2_set[i],
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'
);
END_REPEAT;

(*get all property_definition instances which are the .definition of the
property_definition *)

REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

(*compare both lists with property_definition instances which have to be
identical*)

violation := t1_set = t4_set;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

property_definition: the set of all instances of property_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'stability definition' is referenced by one representation with a class id 'stability table' via a property_definition_representation.

5.2.3.28 one_or_more_items_with_class_hydrostatic_properties_for_constant_floating_position_in_representation_w_class_hydrostatic_table

Ensures that an instance of representation with class id 'hydrostatic table' has in its attribute .items one or more representation_item with a class id 'hydrostatic properties for constant floating position'.

```
*)
RULE
one_or_more_items_with_class_hydrostatic_properties_for_constant_floating
_position_in_representation_w_class_hydrostatic_table FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'hydrostatic
table'*)
```

```
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

(*get all instances of representation that have class id 'hydrostatic
table'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'hydrostatic
properties for constant floating position'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic properties for
      constant floating position');

(*get all instances of representation_item that have class id
'hydrostatic properties for constant floating position'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all representation_item instances which are the .items of the
representation instances that have class id 'hydrostatic table'*)

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);

(*compare both lists with representation_item instances and the
intersection has to be greater 0 *)

          t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set* t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'hydrostatic table' has in its attribute .items one or more representation_item with a class id 'hydrostatic properties for constant floating position'.

5.2.3.29 one_or_more_items_with_class_hydrostatic_property_in_representation_w_class_hydrostatic_table

Ensures that an instance of representation with class id 'hydrostatic table' has in its attribute .items one or more representation_item with a class id 'hydrostatic property'.

```
*)
RULE
one_or_more_items_with_class_hydrostatic_property_in_representation_w_cla
ss_hydrostatic_table FOR (representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'hydrostatic
table'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

(*get all instances of representation that have class id 'hydrostatic
table'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'hydrostatic
property'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic property');

(*get all instances of representation_item that have class id
'hydrostatic property'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all representation_item instances which are the .items of the
representation instances that have class id 'hydrostatic table'*)

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);

(*compare both lists with representation_item instances and the
intersection has to be greater 0 *)

          t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
```

```
    violation := (SIZEOF(t3_set* t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id ‘hydrostatic table’ has in its attribute .items one or more representation_item with a class id ‘hydrostatic property’.

5.2.3.30 one_or_more_items_with_class_in_representation_w_class_offset_point_table_model

Ensures that an instance of representation with class id ‘offset point table model’ has in its attribute .items one or more representation_item with a class id ‘section of offset point table’.

```
*)
RULE
one_or_more_items_with_class_in_representation_w_class_offset_point_table
_model FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’offset point table
model’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’offset point table model’);

(*get all instances of representation that have class id ’offset point
table model’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’section of offset
point table’*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’section of offset point
      table’);
```

```
(*get all instances of representation_item that have class id 'section of
offset point table'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all representation_item instances which are the .items of the
representation instances that have class id 'offset point table model'*)

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);

(*compare both lists with representation_item instances and the
intersection has to be greater 0 *)

          t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set* t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'offset point table model' has in its attribute .items one or more representation_item with a class id 'section of offset point table'.

5.2.3.31 one_or_more_items_with_class_in_representation_w_class_offset_point_table_model2

Ensures that an instance of representation with class id 'offset point table model' has in its attribute .items one or more representation_item with a class id 'ship point'.

```
*)
RULE
one_or_more_items_with_class_in_representation_w_class_offset_point_table
_model2 FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'offset point table
model'*)
```

```
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'offset point table model');

(*get all instances of representation that have class id 'offset point
table model'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'ship point'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship point');

(*get all instances of representation_item that have class id 'ship
point'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all representation_item instances which are the .items of the
representation instances that have class id 'offset point table model'*)

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);

(*compare both lists with representation_item instances and the
intersection has to be greater 0 *)

          t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set* t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'offset point table model' has in its attribute .items one or more representation_item with a class id 'ship point'.

5.2.3.32 one_or_more_items_with_class_in_representation_w_class_offset_table_shape_representation

Ensures that an instance of representation with class id ‘offset table shape representation’ has in its attribute .items one or more representation_item with a class id ‘offset point table model’.

```
*)
RULE
one_or_more_items_with_class_in_representation_w_class_offset_table_shape
_representation FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’offset table shape
representation’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’offset table shape
      representation’);

(*get all instances of representation that have class id ’offset table
shape representation’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’offset point table
model’*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’offset point table model’);

(*get all instances of representation_item that have class id ’offset
point table model’*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all representation_item instances which are the .items of the
representation instances that have class id ’offset table shape
representation’*)

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);

(*compare both lists with representation_item instances and the
intersection has to be greater 0 *)

          t3_set := t3_set + t1_set[i].items[j];
```

```
    END_REPEAT;
    violation := (SIZEOF(t3_set* t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'offset table shape representation' has in its attribute .items one or more representation_item with a class id 'offset point table model'.

5.2.3.33 one_or_more_items_with_class_in_representation_w_class_stability_table

Ensures that an instance of representation with class id 'stability table' has in its attribute .items one or more representation_item with a class id 'stability properties for one or more floating position'.

```
*)
RULE
one_or_more_items_with_class_in_representation_w_class_stability_table
FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'stability table'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability table');

(*get all instances of representation that have class id 'stability
table'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'stability
properties for one floating position'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability properties for one
      floating position');
```

```
(*get all instances of representation_item that have class id 'stability
properties for one floating position'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all representation_item instances which are the .items of the
representation instances that have class id 'stability table'*)

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);

(*compare both lists with representation_item instances and the
intersection has to be greater 0 *)

          t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set* t2_set) < 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'stability table' has in its attribute .items one or more representation_item with a class id 'stability properties for one floating position'.

5.2.3.34 one_or_more_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_properties_for_constant_floating_position

Ensures that an instance of compound_representation_item with class id 'hydrostatic properties for constant floating position' has in its attribute .item_element in the list_representation_item for each entry in the arg list one or more representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_or_more_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_propert
ies_for_constant_floating_position FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['hydrostatic property value'];
    violation: BOOLEAN := FALSE;
```

```
END_LOCAL;

(*get all classification_assignment instances with id 'hydrostatic
properties for constant floating position'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic properties for
      constant floating position');

(*get all instances of compound_representation_item that have class id
'hydrostatic properties for constant floating position'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) < 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id 'hydrostatic properties for constant floating position' has in its attribute .item_element in the list_representation_item for each entry in the arg list one or more representation_item whose .name attribute has the value given in the arg list ['hydrostatic property value'].

5.2.3.35 one_rep_item_in_compound_rep_item_if_class_eq_stability_properties_for_one_floating_position

Ensures that an instance of compound_representation_item with class id 'stability properties for one floating position'has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_rep_item_in_compound_rep_item_if_class_eq_stability_properties_for_on
e_floating_position FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
```

```
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['centre of gravity above keel',
'definition of starting floating position'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'stability
properties for one floating position'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'stability properties for one
      floating position');

(*get all instances of compound_representation_item that have class id
'stability properties for one floating position'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

 (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id 'stability properties for one floating position' has in its attribute .item_element in the list_representation_item for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['centre of gravity above keel', 'definition of starting floating position'].

5.2.3.36 four_or_more_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model

Ensures that an instance of representation with class id 'ship moulded form wireframe model'has in its attribute .items for each entry in the arg list four or more representation_item whose .name attribute has the value given in the arg list.

```
*)
```

```
RULE
four_or_more_item_in_representation_if_class_eq_ship_moulded_form_wirefra
me_model FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['mesh definition ship curve'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'ship moulded form
wireframe model'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form wireframe
      model');

(*get all instances of representation that have class id 'ship moulded
form wireframe model'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
     END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) < 4);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of representation with class id 'ship moulded form wireframe model'has in its attribute .items for each entry in the arg list four or more representation_item whose .name attribute has the value given in the arg list ['mesh definition ship curve'].

5.2.3.37 two_or_more_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_class_knot

Ensures that an instance of compound_representation_item with class id 'knot' has in its attribute .item_element in the list_representation_item two or more compound_representation_item with class id 'ship curve'.

```
*)
RULE
two_or_more_compound_rep_item_with_class_ship_curve_in_compound_rep_item_
with_class_knot FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'knot'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'knot');

(*get all instances of compound_representation_item that have class id
'knot'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'ship curve'*)

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve');

(* get all instances of compound_representation_item that have class id
'ship curve'*)

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;

(*iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second
list of compound_representation_item is equal 1*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) < 2);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id ‘knot’ has in its attribute .item_element in the list_representation_item two or more compound_representation_item with class id ‘ship curve‘.

5.2.3.38 valid_prod_def_w_classs_moulded_form_in_prod_def_w_class_ship_moulded_form

Ensures that an instance of product_definition with class id ‘ship moulded form’ that is also a group (group_assignment.role.name = 'equivalence'; typically a subtype of item_structure in the ARM) collects through the group only product_definition instances with the class ‘moulded form’.

```
*)
RULE
valid_prod_def_w_classs_moulded_form_in_prod_def_w_class_ship_moulded_for
m FOR (product_definition, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF PRODUCT_DEFINITION := [];
    t2_set:  SET OF PRODUCT_DEFINITION:= [];
    gr_ass:    SET OF GROUP_ASSIGNMENT := [];
    groups:    SET OF GROUP := [];
    t3_set:  SET OF PRODUCT_DEFINITION:= [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship moulded
form’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship moulded form’);

(* get all instances of product_definition that have class id ’ship
moulded form’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get the groups associated with these product_definitions *)

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
          i.role.name = ’equivalence’);
REPEAT i := 1 TO HIINDEX(gr_ass);
    REPEAT j := 1 TO HIINDEX(gr_ass[i].items); (*should always be just
one*)
          IF (gr_ass[i].items[j] IN t1_set) THEN
                groups := groups + gr_ass.assigned_group;
          END_IF;
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’moulded form’*)
```

```
c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form');

(* get all instances of product_definition that have class id 'moulded
form'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get the non-empty group_assignments having role 'item structure' and
referencing these groups*)

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
          (SIZEOF(i.items) <> 0) AND
          (i.role.name = 'item structure') AND
          (i.assigned_group IN groups));

(*check if there are only product_definition classed as 'moulded form'in
each group_assignment.items*)

REPEAT i := 1 TO HIINDEX(gr_ass);
    REPEAT j := 1 TO HIINDEX(gr_ass[i].items);
          t3_set := t3_set + gr_ass[i].items[j];
    END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'ship moulded form' that is also a group (group_assignment.role.name = 'equivalence'; typically a subtype of item_structure in the ARM) collects through the group only product_definition instances with the class 'moulded form'.

5.2.3.39 valid_prod_def_rel_w_classs_moulded_form_relationship_in_prod_def_w_class_ship_moulded_form

Ensures that an instance of product_definition with class id 'ship moulded form' that is also a group (group_assignment.role.name = 'equivalence'; typically a subtype of item_structure in the ARM) collects through the group only product_definition_relationship instances with the class 'moulded form relationship'.

```
*)
RULE
valid_prod_def_rel_w_classs_moulded_form_relationship_in_prod_def_w_class
_ship_moulded_form FOR (product_definition,
APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF PRODUCT_DEFINITION := [];
    t2_set:  SET OF PRODUCT_DEFINITION_RELATIONSHIP:= [];
    gr_ass:    SET OF GROUP_ASSIGNMENT := [];
    groups:    SET OF GROUP := [];
    t3_set:  SET OF PRODUCT_DEFINITION_RELATIONSHIP:= [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'ship moulded
form'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form');

(* get all instances of product_definition that have class id 'ship
moulded form'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get the groups associated with these product_definitions *)

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
          i.role.name = 'equivalence');
REPEAT i := 1 TO HIINDEX(gr_ass);
    REPEAT j := 1 TO HIINDEX(gr_ass[i].items); (*should always be just
one*)
          IF (gr_ass[i].items[j] IN t1_set) THEN
                groups := groups + gr_ass.assigned_group;
          END_IF;
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id 'moulded form
relationship'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form relationship');

(* get all instances of product_definition_relationship that have class
id 'moulded form relationship'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t2_set := t2_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get the non-empty group_assignments having role 'item structure' and
referencing these groups*)

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
          (SIZEOF(i.items) <> 0) AND
          (i.role.name = 'item structure') AND
```

```
            (i.assigned_group IN groups));

(*check if there are only product_definition_relationship classed as
'moulded form relationship'in each group_assignment.items*)

REPEAT i := 1 TO HIINDEX(gr_ass);
    REPEAT j := 1 TO HIINDEX(gr_ass[i].items);
          t3_set := t3_set + gr_ass[i].items[j];
    END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'ship moulded form' that is also a group (group_assignment.role.name = 'equivalence'; typically a subtype of item_structure in the ARM) collects through the group only product_definition_relationship instances with the class 'moulded form relationship'.

5.2.3.40 prod_def_rel_w_class_moulded_form_relationship_related_prod_def_shall_have_class_moulded_form

Ensures that an instance of product_definition_relationship with class id 'moulded form relationship' references as the .related_product_definition only a product_definition with class id 'moulded form'.

```
*)
RULE
prod_def_rel_w_class_moulded_form_relationship_related_prod_def_shall_hav
e_class_moulded_form FOR (product_definition_relationship,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: SET OF PRODUCT_DEFINITION_RELATIONSHIP := [];
    t2_set:  SET OF PRODUCT_DEFINITION:= [];
    t3_set:  SET OF PRODUCT_DEFINITION:= [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'moulded form
relationship'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form relationship');

(*get all instances of product_definition_relationship that have class id
'moulded form relationship'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```

```
            t1_set := t1_set + c_a_set[i].items[j];
        END_REPEAT;
    END_REPEAT;

    (*get all .related_product_definition attribute instances of the
    product_definition_relationship*)

    REPEAT i := 1 TO HIINDEX(t1_set);
        t2_set := t2_set + t1_set[i].related_product_definition;
    END_REPEAT;

    (*get all classification_assignment instances with id 'moulded form'*)

    c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
            i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form');

    (*get all instances of product_definition that have class id 'moulded
    form'*)

    REPEAT i := 1 TO HIINDEX(c2_a_set);
        REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
            t3_set := t3_set + c2_a_set[i].items[j];
        END_REPEAT;
    END_REPEAT;
    violation := NOT(t2_set <= t3_set);

    WHERE
        WR1: NOT violation;
    END_RULE;
    (*
```

Argument definitions:

product_definition_relationship: the set of all instances of product_definition_relationship.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition_relationship with class id 'moulded form relationship' references as the .related_product_definition only a product_definition with class id 'moulded form'.

5.2.3.41 prod_def_rel_w_class_moulded_form_relationship_relating_prod_def_shall_have_class_moulded_form

Ensures that an instance of product_definition_relationship with class id 'moulded form relationship' references as the .relating_product_definition only a product_definition with class id 'moulded form'.

```
    *)
    RULE
    prod_def_rel_w_class_moulded_form_relationship_relating_prod_def_shall_ha
    ve_class_moulded_form FOR (product_definition_relationship,
    APPLIED_CLASSIFICATION_ASSIGNMENT);

    LOCAL
        c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
        c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
        t1_set: SET OF PRODUCT_DEFINITION_RELATIONSHIP := [];
        t2_set:  SET OF PRODUCT_DEFINITION:= [];
        t3_set:  SET OF PRODUCT_DEFINITION:= [];
        violation: BOOLEAN := FALSE;
```

```
END_LOCAL;

(*get all classification_assignment instances with id 'moulded form
relationship'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form relationship');

(*get all instances of product_definition_relationship that have class id
'moulded form relationship'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all .relating_product_definition attribute instances of the
product_definition_relationship*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := t2_set + t1_set[i].relating_product_definition;
END_REPEAT;

(*get all classification_assignment instances with id 'moulded form'*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form');

(*get all instances of $1 that have class id 'moulded form'*)

REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t3_set := t3_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition_relationship: the set of all instances of product_definition_relationship.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition_relationship with class id 'moulded form relationship' references as the .relating_product_definition only a product_definition with class id 'moulded form'.

5.2.3.42 dsc_rep_item_w_n_user_def_function_if_dsc_rep_item_w_n_function_and_d_user_defined

Ensures that an instance of representation that has a descriptive_representation_item with name 'function' and description 'user defined' to have another descriptive_representation_item with name 'user def function'.

```
*)
RULE
dsc_rep_item_w_n_user_def_function_if_dsc_rep_item_w_n_function_and_d_use
r_defined FOR (REPRESENTATION);

LOCAL
    rep_set:   SET OF representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
          (r.NAME = 'function') AND (r.DESCRIPTION = 'user defined'))) >
      0) AND
      (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
    (r.NAME = 'user def function'))) = 0);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation that has a descriptive_representation_item with name ‘function‘ and description ‘user defined’ has another descriptive_representation_item with name ‘user def function‘.

5.2.3.43 ID_of_extended_property_definition_with_class_versionable_object_pattern

Ensures that an instance of extended_property_definition with class id ‘versionable object’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_extended_property_definition_with_class_versionable_object_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a 'versionable object'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'versionable object') THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of extended_property_definition with class id ‘versionable object’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.44 ID_of_product_with_class_definable_object_pattern

Ensures that an instance of product with class id ‘definable object’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_with_class_definable_object_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’definable object’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’definable object’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product with class id ‘definable object’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.45 ID_of_product_definition_with_class_class_and_statutory_designation_pattern

Ensures that an instance of product_definition with class id ‘class and statutory designation’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_product_definition_with_class_class_and_statutory_designation_patte
rn FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;
```

```
(*find all instances being classified to be a 'class and statutory
designation'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'class and statutory designation')
THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'class and statutory designation' has an .id attribute that matches the pattern '*\:*'.

5.2.3.46 ID_of_product_definition_with_class_definable_object_pattern

Ensures that an instance of product_definition with class id 'definable object' has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_definition_with_class_definable_object_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a 'definable object'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'definable object') THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'definable object' has an .id attribute that matches the pattern '*\:*'.

5.2.3.47 ID_of_product_definition_with_class_global_axis_placement_pattern

Ensures that an instance of product_definition with class id ‘global axis placement’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_definition_with_class_global_axis_placement_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’global axis placement’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’global axis placement’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘global axis placement’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.48 ID_of_product_definition_with_class_owner_designation_pattern

Ensures that an instance of product_definition with class id ‘owner designation’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_definition_with_class_owner_designation_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’owner designation’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’owner designation’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
```

```
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘owner designation’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.49 ID_of_product_definition_with_class_principal_characteristics_pattern

Ensures that an instance of product_definition with class id ‘principal characteristics’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_product_definition_with_class_principal_characteristics_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’principal
characteristics’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’principal characteristics’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘principal characteristics’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.50 ID_of_product_definition_with_class_ship_designation_pattern

Ensures that an instance of product_definition with class id ‘ship designation’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_definition_with_class_ship_designation_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
```

```
END_LOCAL;

(*find all instances being classified to be a 'ship designation'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'ship designation') THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'ship designation' has an .id attribute that matches the pattern '*\:*'.

5.2.3.51 ID_of_product_definition_with_class_shipyard_designation_pattern

Ensures that an instance of product_definition with class id 'shipyard designation' has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_definition_with_class_shipyard_designation_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a 'shipyard designation'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'shipyard designation') THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘shipyard designation’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.52 ID_of_product_definition_relationship_with_class_definable_object_pattern

Ensures that an instance of product_definition_relationship with class id ‘definable object’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_product_definition_relationship_with_class_definable_object_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’definable object’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’definable object’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition_relationship with class id ‘definable object’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.53 ID_of_product_related_product_category_with_class_ship_type_pattern

Ensures that an instance of product_related_product_category with class id ‘ship type’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_product_related_product_category_with_class_ship_type_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’ship type’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’ship type’) THEN
```

```
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_related_product_category with class id 'ship type' has an .id attribute that matches the pattern '*\:*'.

5.2.3.54 ID_of_property_definition_with_class_deck_moulded_form_design_parameter_pattern

Ensures that an instance of property_definition with class id 'deck moulded form design parameter' has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_property_definition_with_class_deck_moulded_form_design_parameter_p
attern FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a 'deck moulded form design
parameter'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'deck moulded form design parameter')
THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'deck moulded form design parameter' has an .id attribute that matches the pattern '*\:*'.

5.2.3.55 ID_of_property_definition_with_class_hydrostatic_definition_pattern

Ensures that an instance of property_definition with class id ‘hydrostatic definition’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_property_definition_with_class_hydrostatic_definition_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’hydrostatic definition’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’hydrostatic definition’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘hydrostatic definition’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.56 ID_of_property_definition_with_class_local_co_ordinate_system_pattern

Ensures that an instance of property_definition with class id ‘local co-ordinate system’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_property_definition_with_class_local_co_ordinate_system_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’local co-ordinate
system’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’local co-ordinate system’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
```

```
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'local co-ordinate system' has an .id attribute that matches the pattern '*\:*'.

5.2.3.57 ID_of_property_definition_with_class_moulded_form_characteristics_definition_pattern

Ensures that an instance of property_definition with class id 'moulded form characteristics definition' has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE
ID_of_property_definition_with_class_moulded_form_characteristics_definit
ion_pattern FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a 'moulded form
characteristics definition'*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = 'moulded form characteristics
definition') THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'moulded form characteristics definition' has an .id attribute that matches the pattern '*\:*'.

5.2.3.58 ID_of_property_definition_with_class_ship_overall_dimensions_pattern

Ensures that an instance of property_definition with class id ‘ship overall dimensions’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_property_definition_with_class_ship_overall_dimensions_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’ship overall dimensions’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’ship overall dimensions’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘ship overall dimensions’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.59 ID_of_property_definition_with_class_spacing_table_pattern

Ensures that an instance of property_definition with class id ‘spacing table’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_property_definition_with_class_spacing_table_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’spacing table’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’spacing table’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
```

```
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘spacing table’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.60 ID_of_property_definition_with_class_stability_definition_pattern

Ensures that an instance of property_definition with class id ‘stability definition’ has an .id attribute that matches the pattern '*\:*'.

```
*)
RULE ID_of_property_definition_with_class_stability_definition_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    clfied_inst : SET OF GENERIC := [];
END_LOCAL;

(*find all instances being classified to be a ’stability definition’*)

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
    IF (APPLIED_CLASSIFICATION_ASSIGNMENT
[i].assigned_classification.name = ’stability definition’) THEN
    clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT
[i].items;
    END_IF;
END_REPEAT;

WHERE
    WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE ’*\:*’))) = 0;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘stability definition’ has an .id attribute that matches the pattern '*\:*'.

5.2.3.61 rep_has_guac

Ensures that an instance of representation has a global_unit_assigned_context if it has representation_items of type value_representation_item and/ or geometric_representation_item.

```
*)
RULE rep_has_guac FOR (REPRESENTATION);

LOCAL
    v_r_i: STRING := ’SHIP_MOULDED_FORM_AIM.VALUE_REPRESENTATION_ITEM’;
    g_r_i: STRING :=
’SHIP_MOULDED_FORM_AIM.GEOMETRIC_REPRESENTATION_ITEM’;
    g_u_a_c:  STRING :=
’SHIP_MOULDED_FORM_AIM.GLOBAL_UNIT_ASSIGNED_CONTEXT’;
    has_guac: BOOLEAN := TRUE;
```

```
END_LOCAL;

REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE has_guac;
    REPEAT j := 1 TO SIZEOF(REPRESENTATION[i].ITEMS) WHILE has_guac;
          IF ((v_r_i IN TYPEOF(REPRESENTATION[i].ITEMS[j])) OR
                (g_r_i IN TYPEOF(REPRESENTATION[i].ITEMS[j]))) THEN
            has_guac := (g_u_a_c IN
          TYPEOF(REPRESENTATION[i].CONTEXT_OF_ITEMS));
          END_IF;
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: has_guac;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation has a global_unit_assigned_context if it has representation_items of type value_representation_item and/ or geometric_representation_item.

5.2.3.62 max_one_version_id_per_versionable_object

Ensures that an instance that is referenced by an applied_identification_assignment whose role is 'version id' has only one reference of this type.

```
*)
RULE max_one_version_id_per_versionable_object FOR
(APPLIED_IDENTIFICATION_ASSIGNMENT);

LOCAL
    version_ids: SET OF APPLIED_IDENTIFICATION_ASSIGNMENT := [];
    versionable_objects: BAG OF GENERIC := [];
    duplicate: BOOLEAN := FALSE;
END_LOCAL;

(*get all applied_identification_assignments whose role is
'version identifier' *)

version_ids := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
          i.ROLE.NAME = 'version identifier');

(*collect all instances that are referenced*)

REPEAT i := 1 TO HIINDEX(version_ids);
    versionable_objects := versionable_objects + version_ids[i].items;
END_REPEAT;

(*search if there are duplicate instances*)

REPEAT i := 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
    REPEAT j := i + 1 TO HIINDEX(versionable_objects) WHILE NOT
duplicate;
        duplicate := versionable_objects[i] :=: versionable_objects[j];
    END_REPEAT;
END_REPEAT;

WHERE
```

```
    WR1: NOT duplicate;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance that is referenced by an applied_identification_assignment whose role is 'version id' has only one reference of this type.

5.2.3.63 id_is_globally_unique

Ensures the globally uniqueness of the attribute .id of all instances with the class id 'definable object'.

```
*)
RULE id_is_globally_unique FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    duplicate: BOOLEAN := FALSE;
    def_obj_cl_ass: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    clfied_inst: SET OF GENERIC := [];
END_LOCAL;

(*get all instances of applied_classification_assignment having a group
with name 'definable object'*)

def_obj_cl_ass := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.assigned_classification.name = 'definable object');

(*get all instances refered to by these classification_assignments *)

REPEAT i := 1 TO HIINDEX(def_obj_cl_ass);
    clfied_inst := clfied_inst + def_obj_cl_ass[i].items;
END_REPEAT;

(*iterate over the instance list and check the uniqueness *)

REPEAT i := 1 TO HIINDEX(clfied_inst) WHILE NOT duplicate;
    REPEAT j := i + 1 TO HIINDEX(clfied_inst) WHILE NOT duplicate;
          IF (clfied_inst[i].id = clfied_inst[j].id) THEN
                duplicate := TRUE;
          END_IF;
    END_REPEAT;
END_REPEAT;

(*there has to be no duplication in the values of the .id attributes*)

WHERE
    WR1: NOT(duplicate);
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures the globally uniqueness of the attribute .id of all instances with the class id 'definable object'.

5.2.3.64 prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_class_notation

Each instance of product_definition with a class id ‘class and statutory designation’ should be referenced by one property_definition with the class id ‘class notation’ via the attribute .definition.

```
*)
RULE
prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w
_class_id_class_notation FOR (PRODUCT_DEFINITION, PROPERTY_DEFINITION,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF PRODUCT_DEFINITION := [];
    t2_set:  SET OF PROPERTY_DEFINITION := [];
    t3_set:  SET OF PROPERTY_DEFINITION := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’class and
statutory designation’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’class and statutory
      designation’);

(*get all instances of PRODUCT_DEFINITION that have class id ’class and
statutory designation’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’class notation ’*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’class notation’);

(*get all instances of PROPERTY_DEFINITION that have class id ’class
notation’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t3_set := t3_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of PRODUCT_DEFINITION in t1_set; get the
PROPERTY_DEFINITION instances that are referencing a PRODUCT_DEFINITION
instance via .definition, filter out those PROPERTY_DEFINITION instances
whose class id is ’class notation’ check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’PROPERTY_DEFINITION’ + ’.’ + ’DEFINITION’);
END_REPEAT;
violation := t3_set <= t2_set;
```

```
WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

property_definition: the set of all instances of property_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that each instance of product_definition with a class id ‘class and statutory designation’ is referenced by one property_definition with the class id ‘class notation’ via the attribute .definition.

5.2.3.65 prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w_class_id_regulation

Each instance of product_definition with a class id ‘class and statutory designation’ should be referenced by one property_definition with the class id ‘regulation’ via the attribute .definition.

```
*)
RULE
prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w
_class_id_regulation FOR (PRODUCT_DEFINITION, PROPERTY_DEFINITION,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF PRODUCT_DEFINITION := [];
    t2_set:  SET OF PROPERTY_DEFINITION := [];
    t3_set:  SET OF PROPERTY_DEFINITION := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’class and
statutory designation’*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’class and statutory
      designation’);

(*get all instances of PRODUCT_DEFINITION that have class id ’class and
statutory designation’*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*get all classification_assignment instances with id ’regulation’*)

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’regulation’);
```

```
(*get all instances of PROPERTY_DEFINITION that have class id
'regulation'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
          t3_set := t3_set + c2_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of PRODUCT_DEFINITION in t1_set; get the
PROPERTY_DEFINITION instances that are referencing a PRODUCT_DEFINITION
instance via .definition, filter out those PROPERTY_DEFINITION instances
whose class id is 'regulation' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION' + '.' + 'DEFINITION');
END_REPEAT;
violation := t3_set <= t2_set;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

product_definition: the set of all instances of product_definition.

property_definition: the set of all instances of property_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that each instance of product_definition with a class id ‘class and statutory designation’ is referenced by one property_definition with the class id ‘regulation’ via the attribute .definition.

5.2.3.66 one_item_in_representation_if_name_eq_appendage_moulded_form_design_parameter

Ensures that an instance of representation with name ‘appendage moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_appendage_moulded_form_design_param
eter FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['appendage length','appendage
breadth','appendage depth','type of appendage'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'appendage
moulded form design parameter'*)

t1_set:= QUERY(temp_rep <* representation |
```

```
            temp_rep.name = 'appendage moulded form design parameter');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name 'appendage moulded form design parameter' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['appendage length','appendage breadth','appendage breadth','type of appendage'].

5.2.3.67 one_item_in_representation_if_name_eq_bottom_moulded_form_design_parameter

Ensures that an instance of representation with name 'bottom moulded form design parameter' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_bottom_moulded_form_design_paramete
r FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['bilge radius','rise of floor','aft end
of flat of bottom','front end of flat of bottom','flat of bottom
breadth','length of flat of bottom breadth','rake of keel'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'bottom
moulded form design parameter'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'bottom moulded form design parameter');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
```

```
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘bottom moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘bilge radius‘,‘rise of floor‘,‘aft end of flat of bottom‘,‘front end of flat of bottom‘,‘flat of bottom breadth‘,‘length of flat of bottom breadth‘,‘rake of keel‘].

5.2.3.68 one_item_in_representation_if_name_eq_bulb_moulded_form _design_parameter

Ensures that an instance of representation with name ‘bulb moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_bulb_moulded_form_design_parameter
FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’bulb length’,’bulb length from
pp’,’bulb breadth’,’bulb breadth pp’,’bulb depth’,’bulb depth pp’,’bulb
frame section area at pp’,’bulb location’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’bulb
moulded form design parameter’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’bulb moulded form design parameter’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘bulb moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘bulb length‘,‘bulb length from pp‘,‘bulb breadth‘,‘bulb breadth pp‘,‘bulb depth‘,‘bulb depth pp‘,‘bulb frame section area at pp‘,‘bulb location‘].

5.2.3.69 one_item_in_representation_if_name_eq_deck_moulded_form _design_parameter

Ensures that an instance of representation with name ‘deck moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_deck_moulded_form_design_parameter
FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['camber','sheer at AP','sheer at FP'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'deck
moulded form design parameter'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'deck moulded form design parameter');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘deck moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘camber‘,‘sheer at AP‘,‘sheer at FP‘].

5.2.3.70 one_item_in_representation_if_name_eq_hull_moulded_form_design_parameter

Ensures that an instance of representation with name ‘hull moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_hull_moulded_form_design_parameter
FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’aft end of parallel midbody at design
draught’, ’front end of parallel midbody at design draught’, ’length of
parallel midbody at design draught’, ’aft end of flat of side’, ’front
end of flat of side’, ’length of flat of side’,’block coefficient’,
’prismatic coefficient’, ’max wetted frame section area’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’hull
moulded form design parameter’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’hull moulded form design parameter’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘hull moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘aft end of parallel midbody at design draught‘, ‘front end of parallel midbody at design draught‘, ‘length of parallel midbody at design draught‘, ‘aft end of flat of side‘, ‘front end of flat of side‘, ‘length of flat of side‘,‘block coefficient‘, ‘prismatic coefficient‘, ‘max wetted frame section area‘].

5.2.3.71 one_item_in_representation_if_name_eq_propeller_moulded_form_design_parameter

Ensures that an instance of representation with name ‘propeller moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_propeller_moulded_form_design_param
eter FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’type of propulsion’, ’propeller
diameter’, ’chord length at 0 7 radius’, ’thickness at 0 7 radius’,
’number of propeller blades’, ’expanded area ratio’, ’hub diameter
ratio’, ’nominal design pitch ratio’, ’type of propeller blades’, ’rake’,
’skew’, ’design sense of rotation’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’propeller
moulded form design parameter’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’propeller moulded form design parameter’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘propeller moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘type of propulsion‘, ‘propeller diameter‘, ‘chord length at 0 7 radius‘, ‘thickness at 0 7 radius‘, ‘number of propeller blades‘, ‘expanded area ratio‘, ‘hub diameter ratio‘, ‘nominal design pitch ratio‘, ‘type of propeller blades‘, ‘rake‘, ‘skew‘, ‘design sense of rotation‘].

5.2.3.72 one_item_in_representation_if_name_eq_rudder_moulded_form_design_parameter

Ensures that an instance of representation with name ‘rudder moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_rudder_moulded_form_design_paramete
r FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’rudder height’, ’rudder mean height’,
’rudder length’, ’rudder mean length’, ’rudder thickness’, ’projected
rudder area’, ’type of the rudder’, ’rudder location’, ’aspect ratio’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’rudder
moulded form design parameter’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’rudder moulded form design parameter’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘rudder moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘rudder height‘, ‘rudder mean height‘, ‘rudder length‘, ‘rudder mean length‘, ‘rudder thickness‘, ‘projected rudder area‘, ‘type of the rudder‘, ‘rudder location‘, ‘aspect ratio‘].

5.2.3.73 one_item_in_representation_if_name_eq_thruster_moulded_form_design_parameter

Ensures that an instance of representation with name ‘thruster moulded form design parameter’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
```

```
RULE
one_item_in_representation_if_name_eq_thruster_moulded_form_design_parame
ter FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['thruster tunnel diameter', 'thruster
tunnel min length', 'thruster tunnel max length', 'geometric thruster
location', 'thruster location'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'thruster
moulded form design parameter'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'thruster moulded form design parameter');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name 'thruster moulded form design parameter' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['thruster tunnel diameter', 'thruster tunnel min length', 'thruster tunnel max length', 'geometric thruster location', 'thruster location'].

5.2.3.74 one_item_in_representation_if_name_eq_class_and_statutory_designation

Ensures that an instance of representation with name 'class and statutory designation' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_class_and_statutory_designation FOR
(representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class number'];
    violation: BOOLEAN := FALSE;
END_LOCAL;
```

```
(*find all instances of representation whose name is equal to 'class and
statutory designation'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'class and statutory designation');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name 'class and statutory designation' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['class number'].

5.2.3.75 one_item_in_representation_if_name_eq_class_notation

Ensures that an instance of representation with name 'class notation' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_name_eq_class_notation FOR
(representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['service area', 'approval required for
oil cargo', 'approval required for loading unloading aground', 'approval
required for unloading grabs'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'class
notation'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'class notation');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
```

```
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘class notation’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘service area‘, ‘approval required for oil cargo‘, ‘approval required for loading unloading aground‘, ‘approval required for unloading grabs‘].

5.2.3.76 one_item_in_representation_if_name_eq_class_parameters

Ensures that an instance of representation with name ‘class parameters’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_name_eq_class_parameters FOR
(representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['length class', 'length solas',
'scantlings draught', 'block coefficient class', 'design speed
ahead','design speed astern','froude number'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'class
parameters'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'class parameters');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘class parameters’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘length class‘, ‘length solas‘, ‘scantlings draught‘, ‘block coefficient class‘, ‘design speed ahead‘,‘design speed astern‘,‘froude number‘].

5.2.3.77 one_item_in_representation_if_name_eq_global_axis_placement

Ensures that an instance of representation with name ‘global axis placement’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_name_eq_global_axis_placement FOR
(representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['global axes and origin', 'after
perpendicular offset', 'orientation'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'global
axis placement'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'global axis placement');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘global axis placement’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [’global axes and origin’, ‘after perpendicular offset‘, ‘orientation’].

5.2.3.78 one_item_in_representation_if_name_eq_local_co_ordinate_system_with_position_reference

Ensures that an instance of representation with name ‘local co-ordinate system with position reference’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_local_co_ordinate_system_with_posit
ion_reference FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’local axes and origin’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’local co-
ordinate system with position reference’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’local co-ordinate system with position
      reference’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘local co-ordinate system with position reference’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [’local axes and origin’].

5.2.3.79 one_item_in_representation_if_name_eq_local_co_ordinate_system

Ensures that an instance of representation with name ‘local co-ordinate system’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_name_eq_local_co_ordinate_system FOR
(representation);

LOCAL
```

```
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['local axes and origin'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'local co-
ordinate system'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'local co-ordinate system');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name 'local co-ordinate system' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list ['local axes and origin'].

5.2.3.80 one_item_in_representation_if_name_eq_moulded_form_characteristics_definition

Ensures that an instance of representation with name 'moulded form characteristics definition' has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_moulded_form_characteristics_defini
tion FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['moulded form outer surface', 'moulded
form displacement'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'moulded
form characteristics definition'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'moulded form characteristics definition');
```

```
(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘moulded form characteristics definition’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘moulded form outer surface‘, ‘moulded form displacement‘].

5.2.3.81 one_item_in_representation_if_name_eq_moulded_form_function_parameters

Ensures that an instance of representation with name ‘moulded form function parameters’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE
one_item_in_representation_if_name_eq_moulded_form_function_parameters
FOR (representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['function'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to 'moulded
form function parameters'*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = 'moulded form function parameters');

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
```

```
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘moulded form function parameters’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘function‘].

5.2.3.82 one_item_in_representation_if_name_eq_principal_characteristics

Ensures that an instance of representation with name ‘principal characteristics’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_name_eq_principal_characteristics FOR
(representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’length between perpendiculars’,
’moulded breadth’, ’moulded depth’, ’design draught’, ’design
deadweight’,’min daught at fp’,’max daught at fp’, ’min daught at
ap’,’max daught at ap’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’principal
characteristics’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’principal characteristics’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘principal characteristics’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘length between perpendiculars‘, ‘moulded breadth‘, ‘moulded depth‘, ‘design draught‘, ‘design deadweight‘,‘min daught at fp‘,‘max daught at fp‘, ‘min daught at ap‘,‘max daught at ap‘].

5.2.3.83 one_item_in_representation_if_name_eq_ship_overall_dimensions

Ensures that an instance of representation with name ‘ship overall dimensions’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list.

```
*)
RULE one_item_in_representation_if_name_eq_ship_overall_dimensions FOR
(representation);

LOCAL
    t1_set:  SET OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [’overall breadth’, ’overall depth’,
’overall length’, ’stem overhang’, ’stern overhang’];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation whose name is equal to ’ship
overall dimensions’*)

t1_set:= QUERY(temp_rep <* representation |
          temp_rep.name = ’ship overall dimensions’);

(*iterate over all representation instances found above; stop, if one of
them has not exactly one rep_item for each name in the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
          rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that an instance of representation with name ‘ship overall dimensions’ has in its attribute .items for each entry in the arg list one representation_item whose .name attribute has the value given in the arg list [‘overall breadth‘, ‘overall depth‘, ‘overall length‘, ‘stem overhang‘, ‘stern overhang‘].

5.2.3.84 zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name _eq_class_notation

Forces the .items attribute of a representation to have zero or one representation_items whose .name attribute has the value ‘ice class notation‘,‘service factor‘ or ‘approval required for heavy cargo‘ if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘class notation’.

```
*)
RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation
FOR (representation);

LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['ice class notation','service factor',
'approval required for heavy cargo'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation which are used by a
property_definition_representation with name equal to 'class notation'*)

reps := QUERY(temp_rep <* representation |
    SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRES
ENTATION')
    | (temp_prop_def_rep.name = 'class notation'))) > 0 );

(*iterate over all representations found above; stop, if one of them has
not exactly one rep_item for each name in the arg_list *)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name = arg_list[j])) > 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that the .items attribute of a representation has zero or one representation_items whose .name attribute has the value ‘ice class notation‘,‘service factor‘ or ‘approval required for heavy cargo‘ if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘class notation’.

5.2.3.85 zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name _eq_hull_moulded_form_design_parameter

Forces the .items attribute of a representation to have zero or one representation_items whose .name attribute has the value given in the arg list if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘hull moulded form design parameter’.

```
*)
RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_hull_moulded_f
orm_design_parameter FOR (representation);

LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['waterline angle of entrance at stern',
'waterline angle of entrance at bow', 'max frame section area location',
'hull length pp', 'hull breadth', 'hull depth', 'hull design draught',
'gunwale radius'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation which are used by a
property_definition_representation with name equal to 'hull moulded form
design parameter'*)

reps := QUERY(temp_rep <* representation |
    SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRES
ENTATION')
    | (temp_prop_def_rep.name = 'hull moulded form design parameter'))) >
0 );

(*iterate over all representations found above; stop, if one of them has
not exactly one rep_item for each name in the arg_list *)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name = arg_list[j])) > 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that the .items attribute of a representation has zero or one representation_items whose .name attribute has the value given in the arg list [‘waterline angle of entrance at stern‘, ‘waterline angle of entrance at bow‘, ‘max frame section area location‘, ‘hull length pp‘, ‘hull breadth‘, ‘hull depth‘, ‘hull design draught‘, ‘gunwale radius‘] if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘hull moulded form design parameter’.

5.2.3.86 zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_moulded_form_function_parameters

Forces the .items attribute of a representation to have zero or one representation_items whose .name attribute has the value ‘user def function‘ if the representation is the .used_representation in a property_definition_representation whose .name attribute is ‘moulded form function parameters’.

```
*)
```

```
RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_moulded_form_f
unction_parameters FOR (representation);

LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['user def function'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation which are used by a
property_definition_representation with name equal to 'moulded form
function parameters'*)

reps := QUERY(temp_rep <* representation |
    SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRES
ENTATION')
    | (temp_prop_def_rep.name = 'moulded form function parameters'))) > 0
);

(*iterate over all representations found above; stop, if one of them has
not exactly one rep_item for each name in the arg_list *)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name = arg_list[j])) > 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that the .items attribute of a representation has zero or one representation_items whose .name attribute has the 'user def function' if the representation is the .used_representation in a property_definition_representation whose .name attribute is 'moulded form function parameters'.

5.2.3.87 zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name _eq_principal_characteristics

Forces the .items attribute of a representation to have zero or one representation_items whose .name attribute has the value 'block coefficient' if the representation is the .used_representation in a property_definition_representation whose .name attribute is 'principal characteristics'.

```
*)
RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_principal_char
acteristics FOR (representation);

LOCAL
    reps:      BAG OF REPRESENTATION := [];
```

```
    arg_list:  LIST OF STRING := ['block coefficient'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*find all instances of representation which are used by a
property_definition_representation with name equal to 'principal
characteristics'*)

reps := QUERY(temp_rep <* representation |
    SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRES
ENTATION')
    | (temp_prop_def_rep.name = 'principal characteristics'))) > 0 );

(*iterate over all representations found above; stop, if one of them has
not exactly one rep_item for each name in the arg_list *)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name = arg_list[j])) > 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that the .items attribute of a representation has zero or one representation_items whose .name attribute has the 'block coefficient' if the representation is the .used_representation in a property_definition_representation whose .name attribute is 'principal characteristics'.

5.2.3.88 zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name _eq_propeller_moulded_form_design_parameter

Forces the .items attribute of a representation to have zero or one representation_items whose .name attribute has the value 'blade mean height' if the representation is the .used_representation in a property_definition_representation whose .name attribute is 'propeller moulded form design parameter'.

```
*)
RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_propeller_moul
ded_form_design_parameter FOR (representation);

LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['blade mean height'];
    violation: BOOLEAN := FALSE;
END_LOCAL;
```

```
(*find all instances of representation which are used by a
property_definition_representation with name equal to 'propeller moulded
form design parameter'*)

reps := QUERY(temp_rep <* representation |
    SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
    'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRES
ENTATION')
    | (temp_prop_def_rep.name = 'propeller moulded form design
parameter'))) > 0 );

(*iterate over all representations found above; stop, if one of them has
not exactly one rep_item for each name in the arg_list *)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
          violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name = arg_list[j])) > 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of representation.

Formal propositions:

WR1: The condition ensures that the .items attribute of a representation has zero or one representation_items whose .name attribute has the 'blade mean height' if the representation is the .used_representation in a property_definition_representation whose .name attribute is 'propeller moulded form design parameter'.

5.2.3.89 comp_rep_item_w_class_id_section_of_offset_point_table_ref_by__one_class_ass_w_role_section_identifier

Ensures that an instance of compound_representation_item with class id 'section of offset point table' is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value 'section identifier'.

```
*)
RULE
comp_rep_item_w_class_id_section_of_offset_point_table_ref_by__one_class_
ass_w_role_section_identifier FOR(compound_representation_item,
applied_classification_assignment);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF compound_representation_item := [];
    t2_set:  SET OF applied_classification_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'section of offset
point table' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
```

```
            i.ASSIGNED_CLASSIFICATION.NAME = 'section of offset point
        table');

(*get all instances of compound_representation_item that have class id
'section of offset point table' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of compound_representation_item in t1_set, get the
applied_classification_assignment instances that are referencing a
compound_representation_item instance via items, filter out those
applied_classification_assignment instances whose attribute role has the
value 'section identifier' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_CLASSIFICATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'section
identifier')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

compound_representation_item: the set of all instances of compound_representation_item.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of compound_representation_item with class id 'section of offset point table' is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value 'section identifier'.

5.2.3.90 prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managing_company

Ensures that an instance of product_definition with class id 'owner designation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'managing company'.

```
*)
RULE
prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managin
g_company FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: BOOLEAN := FALSE;
```

```
END_LOCAL;

(*get all classification_assignment instances with id 'owner designation'
*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'owner designation');

(*get all instances of product_definition that have class id 'owner
designation' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_organization_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_organization_assignment instances whose attribute role has the
value 'managing company' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'managing
company')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_organization_assignment: the set of all instances of applied_organization_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'owner designation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'managing company'.

5.2.3.91 prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_ordering_company

Ensures that an instance of product_definition with class id 'owner designation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'ordering company'.

```
*)
RULE
prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_orderin
g_company FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
```

```
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'owner designation'
*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'owner designation');

(*get all instances of product_definition that have class id 'owner
designation' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_organization_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_organization_assignment instances whose attribute role has the
value 'ordering company' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'ordering
company')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_organization_assignment: the set of all instances of applied_organization_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'owner designation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'ordering company'.

5.2.3.92 prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_company

Ensures that an instance of product_definition with class id 'owner designation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'owning company'.

```
*)
RULE
prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_
```

```
company FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'owner designation'
*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'owner designation');

(*get all instances of product_definition that have class id 'owner
designation' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_organization_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_organization_assignment instances whose attribute role has the
value 'owning company' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'owning
company')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_organization_assignment: the set of all instances of applied_organization_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘owner designation’ is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value ‘owning company’.

5.2.3.93 prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sign

Ensures that an instance of product_definition with class id ‘ship designation’ is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value ‘call sign’.

```
*)
RULE
prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sig
n FOR(product_definition, applied_classification_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_classification_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship designation’
*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship designation’);

(*get all instances of product_definition that have class id ’ship
designation’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_classification_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_classification_assignment instances whose attribute role has the
value ’call sign’ check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’APPLIED_CLASSIFICATION_ASSIGNMENT’ + ’.’ + ’ITEMS’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = ’call
sign’)) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘ship designation’ is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value ‘call sign’.

5.2.3.94 prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_state

Ensures that an instance of product_definition with class id ‘ship designation’ is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value ‘flag state’.

```
*)
RULE
prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_sta
te FOR(product_definition, applied_classification_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_classification_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship designation’
*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship designation’);

(*get all instances of product_definition that have class id ’ship
designation’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_classification_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_classification_assignment instances whose attribute role has the
value ’flag state’ check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’APPLIED_CLASSIFICATION_ASSIGNMENT’ + ’.’ + ’ITEMS’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = ’flag
state’)) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'ship designation' is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value 'flag state'.

5.2.3.95 prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_registration

Ensures that an instance of product_definition with class id 'ship designation' is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value 'port of registration'.

```
*)
RULE
prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_
registration FOR(product_definition, applied_classification_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_classification_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'ship designation'
*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship designation');

(*get all instances of product_definition that have class id 'ship
designation' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_classification_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_classification_assignment instances whose attribute role has the
value 'port of registration' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_CLASSIFICATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'port of
registration')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_classification_assignment: the set of all instances of applied_classification_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id 'ship designation' is referenced by 1 instances of applied_classification_assignment via the attribute .items and where the attribute .role of applied_classification_assignment has the value 'port of registration'.

5.2.3.96 prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_shipyard

Ensures that an instance of product_definition with class id 'shipyard designation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'shipyard'.

```
*)
RULE
prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_ship
yard FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'shipyard
designation'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'shipyard designation');

(*get all instances of product_definition that have class id 'shipyard
designation'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of product_definition in t1_set, get the
applied_organization_assignment instances that are referencing a
product_definition instance via items, filter out those
applied_organization_assignment instances whose attribute role has the
value 'shipyard' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME =
'shipyard')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
```

```
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

product_definition: the set of all instances of product_definition.

applied_organization_assignment: the set of all instances of applied_organization_assignment.

Formal propositions:

WR1: The condition ensures that an instance of product_definition with class id ‘shipyard designation’ is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value ‘shipyard’.

5.2.3.97 prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_appendage_moulded_form_design_parameter

Ensures that an instance of property_definition with class id ‘appendage moulded form design parameter’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘appendage moulded form design parameter’.

```
*)
RULE
prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_pr
op_def_rep_w_name_appendage_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’appendage moulded
form design parameter’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’appendage moulded form design
      parameter’);

(*get all instances of property_definition that have class id ’appendage
moulded form design parameter’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ’appendage moulded form design parameter’ check if their number
equals 1*)
```

```
REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'appendage
moulded form design parameter')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'appendage moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'appendage moulded form design parameter'.

5.2.3.98 prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bottom_moulded_form_design_parameter

Ensures that an instance of property_definition with class id 'bottom moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'bottom moulded form design parameter'.

```
*)
RULE
prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_
def_rep_w_name_bottom_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'bottom moulded
form design parameter' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'bottom moulded form design
      parameter');

(*get all instances of property_definition that have class id 'bottom
moulded form design parameter' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```

```
            t1_set := t1_set + c_a_set[i].items[j];
        END_REPEAT;
    END_REPEAT;

    (*for all instances of property_definition in t1_set, get the
    property_definition_representation instances that are referencing a
    property_definition instance via definition, filter out those
    property_definition_representation instances whose attribute name has the
    value 'bottom moulded form design parameter' check if their number equals
    1*)

    REPEAT i := 1 TO HIINDEX(t1_set);
        t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
    'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
        violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'bottom moulded
    form design parameter')) = 1);
    END_REPEAT;

    WHERE
        WR1: NOT violation;
    END_RULE;
    (*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'bottom moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'bottom moulded form design parameter'.

5.2.3.99 prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_bulb_moulded_form_design_parameter

Ensures that an instance of property_definition with class id 'bulb moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'bulb moulded form design parameter'.

```
    *)
    RULE
    prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_de
    f_rep_w_name_bulb_moulded_form_design_parameter FOR(property_definition,
    property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

    LOCAL
        c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
        t1_set:  SET OF property_definition := [];
        t2_set:  SET OF property_definition_representation := [];
        violation: BOOLEAN := FALSE;
    END_LOCAL;

    (*get all classification_assignment instances with id 'bulb moulded form
    design parameter' *)
```

```
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'bulb moulded form design
      parameter');

(*get all instances of property_definition that have class id 'bulb
moulded form design parameter' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'bulb moulded form design parameter' check if their number equals
1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'bulb moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'bulb moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'bulb moulded form design parameter'.

5.2.3.100 prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_deck_moulded_form_design_parameter

Ensures that an instance of property_definition with class id 'deck moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'deck moulded form design parameter'.

```
*)
RULE
prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_de
f_rep_w_name_deck_moulded_form_design_parameter FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'deck moulded form
design parameter' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'deck moulded form design
      parameter');

(*get all instances of property_definition that have class id 'deck
moulded form design parameter' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'deck moulded form design parameter' check if their number equals
1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'deck moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'deck moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'deck moulded form design parameter'.

5.2.3.101 prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_hull_moulded_form_design_parameter

Ensures that an instance of property_definition with class id ‘hull moulded form design parameter’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘hull moulded form design parameter’.

```
*)
RULE
prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_de
f_rep_w_name_hull_moulded_form_design_parameter FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’hull moulded form
design parameter’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’hull moulded form design
      parameter’);

(*get all instances of property_definition that have class id ’hull
moulded form design parameter’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ’hull moulded form design parameter’ check if their number equals
1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’PROPERTY_DEFINITION_REPRESENTATION’ + ’.’ + ’DEFINITION’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = ’hull moulded
form design parameter’)) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘hull moulded form design parameter’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘hull moulded form design parameter’.

5.2.3.102 prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_propeller_moulded_form_design_parameter

Ensures that an instance of property_definition with class id ‘propeller moulded form design parameter’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘propeller moulded form design parameter’.

```
*)
RULE
prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_pr
op_def_rep_w_name_propeller_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’propeller moulded
form design parameter’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’propeller moulded form design
      parameter’);

(*get all instances of property_definition that have class id ’propeller
moulded form design parameter’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ’propeller moulded form design parameter’ check if their number
equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’PROPERTY_DEFINITION_REPRESENTATION’ + ’.’ + ’DEFINITION’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = ’propeller
moulded form design parameter’)) = 1);
END_REPEAT;
```

```
WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘propeller moulded form design parameter’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘propeller moulded form design parameter’.

5.2.3.103 prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_rudder_moulded_form_design_parameter

Ensures that an instance of property_definition with class id ‘rudder moulded form design parameter’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘rudder moulded form design parameter’.

```
*)
RULE
prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_
def_rep_w_name_rudder_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’rudder moulded
form design parameter’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’rudder moulded form design
      parameter’);

(*get all instances of property_definition that have class id ’rudder
moulded form design parameter’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
```

```
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'rudder moulded form design parameter' check if their number equals
1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'rudder moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'rudder moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'rudder moulded form design parameter'.

5.2.3.104 prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rep_w_name_thruster_moulded_form_design_parameter

Ensures that an instance of property_definition with class id 'thruster moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'thruster moulded form design parameter'.

```
*)
RULE
prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_pro
p_def_rep_w_name_thruster_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'thruster moulded
form design parameter' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'thruster moulded form design
      parameter');
```

```
(*get all instances of property_definition that have class id 'thruster
moulded form design parameter' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'thruster moulded form design parameter' check if their number
equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'thruster
moulded form design parameter')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'thruster moulded form design parameter' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'thruster moulded form design parameter'.

5.2.3.105 prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_society

Ensures that an instance of property_definition with class id 'class notation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'class society'.

```
*)
RULE
prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_socie
ty FOR(property_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;
```

```
(*get all classification_assignment instances with id 'class notation' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'class notation');

(*get all instances of property_definition that have class id 'class
notation' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
applied_organization_assignment instances that are referencing a
property_definition instance via items, filter out those
applied_organization_assignment instances whose attribute role has the
value 'class society' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'class
society')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

applied_organization_assignment: the set of all instances of applied_organization_assignment.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'class notation' is referenced by 1 instances of applied_organization_assignment via the attribute .items and where the attribute .role of applied_organization_assignment has the value 'class society'.

5.2.3.106 prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_notation

Ensures that an instance of property_definition with class id 'class notation' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'class notation'.

```
*)
RULE
prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_n
otation FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
```

```
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'class notation' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'class notation');

(*get all instances of property_definition that have class id 'class
notation' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'class notation' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'class
notation')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'class notation' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'class notation'.

5.2.3.107 prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_reference

Ensures that an instance of property_definition with class id 'local co-ordinate system with position reference' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'local co-ordinate system with position reference'.

```
*)
RULE
prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_
by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_referen
```

```
ce FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'local co-ordinate
system with position reference' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'local co-ordinate system with
      position reference');

(*get all instances of property_definition that have class id 'local co-
ordinate system with position reference' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'local co-ordinate system with position reference' check if their
number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'local co-
ordinate system with position reference')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'local co-ordinate system with position reference' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'local co-ordinate system with position reference'.

5.2.3.108 prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_name_local_co_ordinate_system

Ensures that an instance of property_definition with class id ‘local co-ordinate system’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘local co-ordinate system’.

```
*)
RULE
prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_na
me_local_co_ordinate_system FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’local co-ordinate
system’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’local co-ordinate system’);

(*get all instances of property_definition that have class id ’local co-
ordinate system’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ’local co-ordinate system’ check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’PROPERTY_DEFINITION_REPRESENTATION’ + ’.’ + ’DEFINITION’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = ’local co-
ordinate system’)) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘local co-ordinate system’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘local co-ordinate system’.

5.2.3.109 prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition

Ensures that an instance of property_definition with class id ‘moulded form characteristics definition’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘moulded form characteristics definition’.

```
*)
RULE
prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_pr
op_def_rep_w_name_moulded_form_characteristics_definition
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’moulded form
characteristics definition’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’moulded form characteristics
      definition’);

(*get all instances of property_definition that have class id ’moulded
form characteristics definition’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ’moulded form characteristics definition’ check if their number
equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’PROPERTY_DEFINITION_REPRESENTATION’ + ’.’ + ’DEFINITION’);
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = ’moulded form
characteristics definition’)) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
```

```
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id ‘moulded form characteristics definition’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘moulded form characteristics definition’.

5.2.3.110 prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_prop_def_rep_w_name_moulded_form_characteristics_definition

Ensures that an instance of property_definition with class id ‘ship overall dimensions’ is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value ‘ship overall dimensions’.

```
*)
RULE
prop_def_w_class_id_ship_overall_dimensions_ref_by_one_prop_def_rep_w_nam
e_ship_overall_dimensions FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id ’ship overall
dimensions’ *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = ’ship overall dimensions’);

(*get all instances of property_definition that have class id ’ship
overall dimensions’ *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ’ship overall dimensions’ check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], ’SHIP_MOULDED_FORM_AIM.’ +
’PROPERTY_DEFINITION_REPRESENTATION’ + ’.’ + ’DEFINITION’);
```

```
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'ship overall
dimensions')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'ship overall dimensions' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'ship overall dimensions'.

5.2.3.111 prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_transformation_w_name_transformation_to_parent

Ensures that an instance of property_definition with class id 'local co-ordinate system' is referenced by 1 instances of item_defined_transformation via the attribute .transform_item_1 and where the attribute .name of item_defined_transformation has the value 'transformation to parent'.

```
*)
RULE
prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_tran
sformation_w_name_transformation_to_parent FOR(property_definition,
item_defined_transformation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF item_defined_transformation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'local co-ordinate
system' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'local co-ordinate system');

(*get all instances of property_definition that have class id 'local co-
ordinate system' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
item_defined_transformation instances that are referencing a
property_definition instance via transform_item_1, filter out those
```

```
item_defined_transformation instances whose attribute name has the value
'transformation to parent' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'ITEM_DEFINED_TRANSFORMATION' + '.' + 'TRANSFORM_ITEM_1');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'transformation
to parent')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

item_defined_transformation: the set of all instances of item_defined_transformation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'local co-ordinate system' is referenced by 1 instances of item_defined_transformation via the attribute .transform_item_1 and where the attribute .name of item_defined_transformation has the value 'transformation to parent'.

5.2.3.112 prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_def_rep_w_name_moulded_form_function_parameters

Ensures that an instance of property_definition with class id 'moulded form functional definition' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'moulded form function parameters'.

```
*)
RULE
prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_de
f_rep_w_name_moulded_form_function_parameters FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

(*get all classification_assignment instances with id 'moulded form
functional definition' *)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form functional
      definition');

(*get all instances of property_definition that have class id 'moulded
form functional definition' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```

```
            t1_set := t1_set + c_a_set[i].items[j];
        END_REPEAT;
    END_REPEAT;

    (*for all instances of property_definition in t1_set, get the
    property_definition_representation instances that are referencing a
    property_definition instance via definition, filter out those
    property_definition_representation instances whose attribute name has the
    value 'moulded form function parameters' check if their number equals 1*)

    REPEAT i := 1 TO HIINDEX(t1_set);
        t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
    'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');
        violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'moulded form
    function parameters')) = 1);
    END_REPEAT;

    WHERE
        WR1: NOT violation;
    END_RULE;
    (*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_representation: the set of all instances of property_definition_representation.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'moulded form functional definition' is referenced by 1 instances of property_definition_representation via the attribute .definition and where the attribute .name of property_definition_representation has the value 'moulded form function parameters'.

5.2.3.113 prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_prop_def_rel_w_name_thruster_propeller_parameter

Ensures that an instance of property_definition with class id 'thruster moulded form design parameter' is referenced by 1 instances of property_definition_relationship via the attribute .relating_property_definition and where the attribute .name of property_definition_relationship has the value 'thruster propeller parameter'.

```
    *)
    RULE
    prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_pro
    p_def_rel_w_name_thruster_propeller_parameter FOR(property_definition,
    property_definition_relationship, APPLIED_CLASSIFICATION_ASSIGNMENT);

    LOCAL
        c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
        t1_set:  SET OF property_definition := [];
        t2_set:  SET OF property_definition_relationship := [];
        violation: BOOLEAN := FALSE;
    END_LOCAL;

    (*get all classification_assignment instances with id 'thruster moulded
    form design parameter' *)

    c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
```

```
            i.ASSIGNED_CLASSIFICATION.NAME = 'thruster moulded form design
        parameter');

(*get all instances of property_definition that have class id 'thruster
moulded form design parameter' *)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(*for all instances of property_definition in t1_set, get the
property_definition_relationship instances that are referencing a
property_definition instance via relating_property_definition, filter out
those property_definition_relationship instances whose attribute name has
the value 'thruster propeller parameter' check if their number equals 1*)

REPEAT i := 1 TO HIINDEX(t1_set);
    t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_RELATIONSHIP' + '.' +
'RELATING_PROPERTY_DEFINITION');
    violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'thruster
propeller parameter')) = 1);
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of applied_classification_assignment.

property_definition: the set of all instances of property_definition.

property_definition_relationship: the set of all instances of property_definition_relationship.

Formal propositions:

WR1: The condition ensures that an instance of property_definition with class id 'thruster moulded form design parameter' is referenced by 1 instances of property_definition_relationship via the attribute .relating_property_definition and where the attribute .name of property_definition_relationship has the value 'thruster propeller parameter'.

5.2.3.114 components_related_to_address

Ensures that each element in the parameter list is used in the description attribute of an external_source, which shall be related to another external_source of class 'address'.

```
*)
RULE components_related_to_address FOR (EXTERNAL_SOURCE);

LOCAL
    parm_list: LIST OF STRING := ['internal location', 'street number',
'street', 'postal box', 'town', 'region', 'postal code', 'country',
'facsimile number', 'telephone number', 'electronic mail address',
'telex number'];
    addr_list: SET OF EXTERNAL_SOURCE;
    comp_list: SET OF EXTERNAL_SOURCE;
    addr_relationships: SET OF EXTERNAL_SOURCE_RELATIONSHIP := [];
    comp_relationships: SET OF EXTERNAL_SOURCE_RELATIONSHIP := [];
    found: BOOLEAN := FALSE;
```

```
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
END_LOCAL;

(*get all classification_assignment instances with id 'address'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'address');

(*get all instances of EXTERNAL_SOURCE that have class id 'address'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          addr_list := addr_list + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

(* find all instances of external_source_relationship in which these
external_sources are used as related_source*)

REPEAT i:=1 TO HIINDEX(addr_list);
    addr_relationships := addr_relationships +
USEDIN(addr_list[i],'SHIP_MOULDED_FORM_AIM.EXTERNAL_SOURCE_RELATIONSHIP.R
ELATED_SOURCE');
END_REPEAT;

(* For each parameter: find all instances of external_source with
description attribute equal to this parameter
*)

REPEAT i:=1 TO HIINDEX(parm_list) WHILE (NOT found);
    comp_list := QUERY(temp <* EXTERNAL_SOURCE
          | temp.description = parm_list[i] );
END_REPEAT;

(* search whether there is an external_source_relationship that relates
an external_sources as image of an address component to another
external_source, that represents the address
*)

REPEAT j:=1 TO HIINDEX(comp_list) WHILE (NOT found);
    comp_relationships := USEDIN(comp_list[j],
'SHIP_MOULDED_FORM_AIM.EXTERNAL_SOURCE_RELATIONSHIP.RELATING_SOURCE');
    found := SIZEOF(addr_relationships * comp_relationships) = 0 ;
END_REPEAT;

WHERE
    WR1: NOT found;
END_RULE;
(*
```

Argument definitions:

external_source: the set of all instances of external_source.

Formal propositions:

WR1: The condition ensures that each element in the parameter list [‘internal location’, ‘street number’, ‘street’, ‘postal box’, ‘town’, ‘region’, ‘postal code’, ‘country’, ‘facsimile number’, ‘telephone number’, ‘electronic mail address’, ‘telex number’] is used in the description attribute of an external_source, which shall be related to another external_source of class ‘address’.

5.2.4 Ship moulded function definitions

5.2.4.1 get_extended_property_definition_id_value

The function get_extended_property_definition_id_value finds the global id provided by an applied_identification_assignment for an extended_property_definition.

```
*)
FUNCTION get_extended_property_definition_id_value
(obj : extended_property_definition) : identifier;
LOCAL
c_a_set: SET OF applied_identification_assignment := [];
t1_set:  SET OF extended_property_definition:= [];
global_id : identifier;
END_LOCAL;

(*get all identification_assignment instances with role.name 'global
id'*)

c_a_set := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
          i.ROLE.NAME = 'global id');

(*get all instances of extended_property_definition that are in the
.items list*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
          IF obj :=: c_a_set[i].items[j] THEN
                global_id := c_a_set[i].assigned_id;
          END_IF;
    END_REPEAT;
END_REPEAT;
IF obj IN t1_set
    THEN RETURN (global_id);
    ELSE RETURN (?);
END_IF;
END_FUNCTION;
(*
```

5.2.4.2 get_extended_product_definition_shape_id_value

The function get_extended_product_definition_shape_id_value finds the global id provided by an applied_identification_assignment for an extended_product_definition_shape.

```
*)
FUNCTION get_extended_product_definition_shape_id_value
(obj : extended_product_definition_shape) : identifier;
LOCAL
c_a_set: SET OF applied_identification_assignment := [];
t1_set:  SET OF extended_product_definition_shape := [];
global_id : identifier;
END_LOCAL;

(*get all identification_assignment instances with role.name 'global
id'*)

c_a_set := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
          i.ROLE.NAME = 'global id');

(*get all instances of extended_product_definition_shape that are in the
.items list*)
```

```
REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
          IF obj :=: c_a_set[i].items[j] THEN
                global_id := c_a_set[i].assigned_id;
          END_IF;
    END_REPEAT;
END_REPEAT;
IF obj IN t1_set
    THEN RETURN (global_id);
    ELSE RETURN (?);
END_IF;
END_FUNCTION;
(*
```

5.2.4.3 get_extended_product_related_product_category_id_value

The function get_extended_product_related_product_category_id_value finds the global id provided by an applied_identification_assignment for an extended_product_related_product_category.

```
*)
FUNCTION get_extended_product_related_product_category_id_value
(obj : extended_product_related_product_category) : identifier;
LOCAL
c_a_set: SET OF applied_identification_assignment := [];
t1_set:  SET OF extended_product_related_product_category := [];
global_id : identifier;
END_LOCAL;

(*get all identification_assignment instances with role.name 'global
id'*)

c_a_set := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
          i.ROLE.NAME = 'global id');

(*get all instances of extended_product_related_product_category that are
in the .items list*)

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
          IF obj :=: c_a_set[i].items[j] THEN
                global_id := c_a_set[i].assigned_id;
          END_IF;
    END_REPEAT;
END_REPEAT;
IF obj IN t1_set
    THEN RETURN (global_id);
    ELSE RETURN (?);
END_IF;
END_FUNCTION;


END_SCHEMA; -- ship_moulded_form_AIM
```

(*

<u>Problems:</u>

— in the 3 AP 216 functions applied_classification_assignment not visible

— error in rules 5.2.3.62 and 5.2.3.63; type GENERIC not possible, because not all entities have an attribute .id.

— errors in the part 40ies

*)

6. Conformance requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, the requirements of the implementation methods supported, and the relevant requirements of the normative references.

An implementation shall support at least one of the following implementation methods:

— ISO 10303-21;

— ISO 10303-22.

Requirements with respect to implementation methods- specific requirements are specified in annex C.

The Protocol Information Conformance Statement (PICS) proforma lists the options or combinations of options that may be included in the implementation. The PICS proforma is provided in annex D.

This part of ISO 10303 provides a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes:

— Class 1 is a conformance class to exchange hydrostatic data

— Class 2 is a conformance class to exchange moulded form geometry as an offset table

— Class 3 is a conformance class to exchange moulded form geometry as a wireframe representation

— Class 4 is a conformance class to exchange moulded form geometry as a surface representation

Conformance to a particular class requires that all AIM elements defined as part of that class be supported. Table 1 defines the classes to which each AIM element belongs.

NOTE - ISO 10303-316 defines the abstract test suite to be used in the assesment of conformance[3]. ISO 10303-32 describes the conformance assesment process.

Unit of Functionality	Conformance class			
	Class 1	Class 2	Class 3	Class 4
basic_geometry		X	X	X
configuration_management	X	X	X	X
date_time_resources	X	X	X	X
definitions	X	X	X	X
external_references	X	X	X	X
hydrostatics	X			
items	X	X	X	X
location_concepts	X	X	X	X
measures_resources	X	X	X	X
offset_table_representations		X		
organisation_resources	X	X	X	X
representations	X	X	X	X
shape	X	X	X	X

[3] Not yet IS

ship_design_parameter	X	X	X	X
ship_general_characteristics	X	X	X	X
ship_measures	X	X	X	X
ship_moulded_forms	X	X	X	X
surface_representations				X
wireframe_representations			X	

Table 1 - Conformance classes

Annex A
(normative)

AIM EXPRESS expanded listing

The following EXPRESS is the expanded form of the short form schema given in 5.2. In the event of discrepancy between the short form and this expanded listing, the expanded listing shall be used.

```
SCHEMA ship_moulded_form_AIM;


TYPE classification_item = SELECT
(product_definition,
product_definition_relationship,
product_definition_shape,
group,
extended_property_definition,
extended_product_definition_shape,
shape_representation,
representation_relationship,
property_definition,
product,
extended_product_related_product_category,
representation,
compound_representation_item,
axis2_placement_3d,
edge_curve,
vertex_point,
measure_with_unit);
END_TYPE;

TYPE document_reference_item = SELECT
(property_definition);
END_TYPE;

TYPE group_item = SELECT
(product_definition,
product_definition_shape,
product_definition_relationship);
END_TYPE;

TYPE organization_item = SELECT
(product_definition,
property_definition);
END_TYPE;

TYPE identification_item = SELECT
(product_definition,
extended_property_definition,
extended_product_definition_shape,
extended_product_related_product_category,
compound_representation_item,
product_definition_relationship,
group);
END_TYPE;

TYPE external_identification_item = SELECT
(product,
product_definition);
END_TYPE;
```

```
TYPE action_request_item = SELECT
(action,
executed_action);
END_TYPE;

TYPE action_item = SELECT
(action_request_solution,
executed_action,
product_definition,
product_definition_relationship,
product_definition_shape,
group,
property_definition,
product,
product_related_product_category);
END_TYPE;

TYPE approval_item = SELECT
(product_definition,
product_definition_shape,
property_definition,
product_related_product_category);
END_TYPE;

TYPE person_and_organization_item = SELECT
(action_request_solution,
executed_action,
versioned_action_request,
action);
END_TYPE;

TYPE date_and_time_item = SELECT
(action_request_solution,
executed_action,
versioned_action_request,
action);
END_TYPE;

  TYPE length_measure = REAL;
  END_TYPE; -- length_measure

  TYPE positive_length_measure = length_measure;
  WHERE
    WR1: SELF > 0;
  END_TYPE; -- positive_length_measure

  TYPE ratio_measure = REAL;
  END_TYPE; -- ratio_measure

  TYPE plane_angle_measure = REAL;
  END_TYPE; -- plane_angle_measure

  TYPE parameter_value = REAL;
  END_TYPE; -- parameter_value

  TYPE area_measure = REAL;
  END_TYPE; -- area_measure

TYPE list_representation_item = LIST [1:?] OF representation_item;
END_TYPE;
```

```
    TYPE label = STRING;
    END_TYPE; -- label

    TYPE text = STRING;
    END_TYPE; -- text

    TYPE identifier = STRING;
    END_TYPE; -- identifier

    TYPE id_attribute_select = SELECT
      (action,
       application_context,
       organizational_project,
       representation);
    END_TYPE; -- id_attribute_select

  TYPE DESCRIPTION_ATTRIBUTE_SELECT = SELECT (
     ACTION_REQUEST_SOLUTION,
     APPLICATION_CONTEXT,
     DATE_TIME_ROLE,
     EXTERNAL_SOURCE,
     ORGANIZATION_ROLE,
     PERSON_AND_ORGANIZATION_ROLE,
     PERSON_AND_ORGANIZATION,
     PROPERTY_DEFINITION_REPRESENTATION,
     REPRESENTATION);
  END_TYPE;

  TYPE NAME_ATTRIBUTE_SELECT = SELECT (
     ACTION_REQUEST_SOLUTION,
     PERSON_AND_ORGANIZATION,
     PRODUCT_DEFINITION,
     PROPERTY_DEFINITION_REPRESENTATION);
  END_TYPE;

  TYPE ROLE_SELECT = SELECT (
     ACTION_ASSIGNMENT,
     ACTION_REQUEST_ASSIGNMENT,
     APPROVAL_ASSIGNMENT,
     DOCUMENT_REFERENCE,
     GROUP_ASSIGNMENT);
  END_TYPE;

    TYPE year_number = INTEGER;
    END_TYPE; -- year_number

    TYPE hour_in_day = INTEGER;
    WHERE
      WR1: { 0 <= SELF < 24 };
    END_TYPE; -- hour_in_day

    TYPE minute_in_hour = INTEGER;
    WHERE
      WR1: { 0 <= SELF <= 59 };
    END_TYPE; -- minute_in_hour

    TYPE second_in_minute = REAL;
    WHERE
      WR1: { 0 <= SELF <= 60 };
    END_TYPE; -- second_in_minute
```

```
 TYPE ahead_or_behind = ENUMERATION OF
   (ahead,
    exact,
    behind);
 END_TYPE; -- ahead_or_behind

TYPE b_spline_curve_form = ENUMERATION OF
  (polyline_form,
   circular_arc,
   elliptic_arc,
   parabolic_arc,
   hyperbolic_arc,
   unspecified);
END_TYPE;

TYPE pcurve_or_surface = SELECT
  (pcurve,
   surface);
END_TYPE;

TYPE preferred_surface_curve_representation = ENUMERATION OF
  (curve_3d,
   pcurve_s1,
   pcurve_s2);
END_TYPE;

TYPE curve_on_surface = SELECT
  (pcurve,
   surface_curve,
   composite_curve_on_surface);
END_TYPE;

TYPE transition_code = ENUMERATION OF
  (discontinuous,
   continuous,
   cont_same_gradient,
   cont_same_gradient_same_curvature);
END_TYPE;

TYPE vector_or_direction = SELECT
  (vector,
   direction);
END_TYPE;

TYPE b_spline_surface_form = ENUMERATION OF
  (plane_surf,
   cylindrical_surf,
   conical_surf,
   spherical_surf,
   toroidal_surf,
   surf_of_revolution,
   ruled_surf,
   generalised_cone,
   quadric_surf,
   surf_of_linear_extrusion,
   unspecified);
END_TYPE;

TYPE AXIS2_PLACEMENT = SELECT (
  AXIS2_PLACEMENT_3D);
```

```
END_TYPE;

TYPE CHARACTERIZED_PRODUCT_DEFINITION = SELECT (
   PRODUCT_DEFINITION,
   PRODUCT_DEFINITION_RELATIONSHIP);
END_TYPE;

TYPE CHARACTERIZED_DEFINITION = SELECT (
   CHARACTERIZED_PRODUCT_DEFINITION,
   SHAPE_DEFINITION);
END_TYPE;

TYPE REPRESENTED_DEFINITION = SELECT (
   PROPERTY_DEFINITION,
   PROPERTY_DEFINITION_RELATIONSHIP,
   SHAPE_ASPECT);
END_TYPE;

TYPE dimension_count = INTEGER;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE SHELL = SELECT (
   VERTEX_SHELL,
   WIRE_SHELL);
END_TYPE;

TYPE SOURCE_ITEM = SELECT (
   IDENTIFIER);
END_TYPE;

  TYPE si_prefix = ENUMERATION OF
    (atto,
     centi,
     deca,
     deci,
     exa,
     femto,
     giga,
     hecto,
     kilo,
     mega,
     micro,
     milli,
     nano,
     peta,
     pico,
     tera);
  END_TYPE; -- si_prefix

  TYPE si_unit_name = ENUMERATION OF
    (ampere,
     becquerel,
     candela,
     coulomb,
     degree_Celsius,
     farad,
     gram,
     gray,
     henry,
```

```
     hertz,
     joule,
     kelvin,
     lumen,
     lux,
     metre,
     mole,
     newton,
     ohm,
     pascal,
     radian,
     second,
     steradian,
     tesla,
     volt,
     watt,
     siemens,
     sievert,
     weber);
  END_TYPE; -- si_unit_name

TYPE UNIT = SELECT (
   NAMED_UNIT);
END_TYPE;

TYPE MEASURE_VALUE = SELECT (
   AREA_MEASURE,
   LENGTH_MEASURE,
   PARAMETER_VALUE,
   PLANE_ANGLE_MEASURE,
   POSITIVE_LENGTH_MEASURE,
   RATIO_MEASURE);
END_TYPE;

TYPE COMPOUND_ITEM_DEFINITION = SELECT (
   LIST_REPRESENTATION_ITEM);
END_TYPE;

TYPE SUPPORTED_ITEM = SELECT (
   ACTION,
   ACTION_METHOD);
END_TYPE;

TYPE SHAPE_DEFINITION = SELECT (
   PRODUCT_DEFINITION_SHAPE,
   SHAPE_ASPECT);
END_TYPE;

TYPE DERIVED_PROPERTY_SELECT = SELECT (
   PROPERTY_DEFINITION);
END_TYPE;

  TYPE person_organization_select = SELECT
    (person,
     organization,
     person_and_organization);
  END_TYPE; -- person_organization_select

 TYPE trimming_select = SELECT
   (cartesian_point,
    parameter_value);
```

```
 END_TYPE;

 TYPE geometric_set_select = SELECT
   (point,
    curve,
    surface);
 END_TYPE;

TYPE WIREFRAME_MODEL = SELECT (
   SHELL_BASED_WIREFRAME_MODEL);
END_TYPE;

TYPE SURFACE_MODEL = SELECT (
   FACE_BASED_SURFACE_MODEL);
END_TYPE;

TYPE transformation = SELECT
  (item_defined_transformation,
   functionally_defined_transformation);
END_TYPE;

  TYPE date_time_select = SELECT
    (date,
     local_time,
     date_and_time);
  END_TYPE; -- date_time_select

TYPE REVERSIBLE_TOPOLOGY_ITEM = SELECT (
   EDGE,
   PATH,
   FACE,
   FACE_BOUND);
END_TYPE;

TYPE DATE_TIME_OR_EVENT_OCCURRENCE = SELECT (
   DATE_TIME_SELECT);
END_TYPE;

TYPE REVERSIBLE_TOPOLOGY = SELECT (
   REVERSIBLE_TOPOLOGY_ITEM);
END_TYPE;

ENTITY applied_classification_assignment
SUBTYPE OF (classification_assignment);
items : SET [1:?] OF classification_item;
END_ENTITY;

ENTITY applied_document_reference
SUBTYPE OF (document_reference);
items : SET [1:?] OF document_reference_item;
END_ENTITY;

ENTITY applied_group_assignment
SUBTYPE OF (group_assignment);
items : SET [1:?] OF group_item;
END_ENTITY;

ENTITY applied_external_identification_assignment
SUBTYPE OF (external_identification_assignment);
items : SET [1:?] OF external_identification_item;
END_ENTITY;
```

```
ENTITY applied_identification_assignment
SUBTYPE OF (identification_assignment);
items : SET [1:?] OF identification_item;
END_ENTITY;

ENTITY applied_organization_assignment
SUBTYPE OF (organization_assignment);
items : SET [1:?] OF organization_item;
END_ENTITY;

ENTITY applied_action_request_assignment
SUBTYPE OF (action_request_assignment);
items : SET [1:?] OF action_request_item;
END_ENTITY;

ENTITY applied_action_assignment
SUBTYPE OF (action_assignment);
items : SET [1:?] OF action_item;
END_ENTITY;

ENTITY applied_approval_assignment
SUBTYPE OF (approval_assignment);
items : SET [1:?] OF approval_item;
END_ENTITY;

ENTITY applied_person_and_organization_assignment
SUBTYPE OF (person_and_organization_assignment);
items : SET [1:?] OF person_and_organization_item;
END_ENTITY;

ENTITY applied_date_and_time_assignment
SUBTYPE OF (date_and_time_assignment);
items : SET [1:?] OF date_and_time_item;
END_ENTITY;

ENTITY class
SUBTYPE OF (group);
END_ENTITY;

ENTITY extended_property_definition
SUBTYPE OF (property_definition);
DERIVE
id: identifier := get_extended_property_definition_id_value(SELF);
END_ENTITY;

ENTITY extended_product_definition_shape
SUBTYPE OF (product_definition_shape);
DERIVE
id: identifier := get_extended_product_definition_shape_id_value(SELF);
END_ENTITY;

ENTITY extended_product_related_product_category
SUBTYPE OF (product_related_product_category);
DERIVE
id: identifier :=
get_extended_product_related_product_category_id_value(SELF);
END_ENTITY;

  ENTITY approval;
    status : approval_status;
```

```
  level  : label;
END_ENTITY; -- approval

ENTITY approval_status;
  name : label;
END_ENTITY; -- approval_status

ENTITY action;
  name          : label;
  description   : OPTIONAL text;
  chosen_method : action_method;
DERIVE
  id            : identifier := get_id_value (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- action

ENTITY executed_action
  SUBTYPE OF (action);
END_ENTITY; -- executed_action

ENTITY action_method;
  name        : label;
  description : OPTIONAL text;
  consequence : text;
  purpose     : text;
END_ENTITY; -- action_method

ENTITY action_request_solution;
  method      : action_method;
  request     : versioned_action_request;
DERIVE
  description : text := get_description_value (SELF);
  name        : label := get_name_value (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
  WR2 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- action_request_solution

ENTITY action_relationship;
  name            : label;
  description     : OPTIONAL text;
  relating_action : action;
  related_action  : action;
END_ENTITY; -- action_relationship

ENTITY versioned_action_request;
  id          : identifier;
  version     : label;
  purpose     : text;
  description : OPTIONAL text;
END_ENTITY; -- versioned_action_request

ENTITY classification_assignment
  ABSTRACT SUPERTYPE;
  assigned_classification : group;
  role                    : classification_role;
END_ENTITY; -- classification_assignment
```

```
ENTITY group_assignment
  ABSTRACT SUPERTYPE;
  assigned_group : group;
DERIVE
  role           : object_role := get_role (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- group_assignment

ENTITY identification_assignment
  ABSTRACT SUPERTYPE;
  assigned_id : identifier;
  role        : identification_role;
END_ENTITY; -- identification_assignment

ENTITY external_identification_assignment
  ABSTRACT SUPERTYPE
SUBTYPE OF (identification_assignment);
  source : external_source;
END_ENTITY; -- external_identification_assignment

ENTITY organization_assignment
  ABSTRACT SUPERTYPE;
  assigned_organization : organization;
  role                  : organization_role;
END_ENTITY; -- organization_assignment

ENTITY document_reference
  ABSTRACT SUPERTYPE;
  assigned_document : document;
  source            : label;
DERIVE
  role              : object_role := get_role (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- document_reference

ENTITY action_assignment
  ABSTRACT SUPERTYPE;
  assigned_action : action;
DERIVE
  role            : object_role := get_role (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- action_assignment

ENTITY action_request_assignment
  ABSTRACT SUPERTYPE;
  assigned_action_request : versioned_action_request;
DERIVE
  role                    : object_role := get_role (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- action_request_assignment

ENTITY approval_assignment
```

```
      ABSTRACT SUPERTYPE;
      assigned_approval : approval;
    DERIVE
      role              : object_role := get_role (SELF);
    WHERE
      WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                           'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
    END_ENTITY; -- approval_assignment

    ENTITY date_and_time_assignment
      ABSTRACT SUPERTYPE;
      assigned_date_and_time : date_and_time;
      role                   : date_time_role;
    END_ENTITY; -- date_and_time_assignment

    ENTITY person_and_organization_assignment
      ABSTRACT SUPERTYPE;
      assigned_person_and_organization : person_and_organization;
      role                             : person_and_organization_role;
    END_ENTITY; -- person_and_organization_assignment

  ENTITY non_manifold_surface_shape_representation
     SUBTYPE OF (shape_representation);
  WHERE
    WR1: SIZEOF (QUERY (it <* SELF.items |
         NOT (SIZEOF (['SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL',
         'SHIP_MOULDED_FORM_AIM.MAPPED_ITEM',
         'SHIP_MOULDED_FORM_AIM.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1)))
  = 0;
    WR2: SIZEOF (QUERY (it <* SELF.items |
         SIZEOF (['SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL',
         'SHIP_MOULDED_FORM_AIM.MAPPED ITEM'] * TYPEOF (it)) = 1)) > 0;
    WR3: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
         'SHIP_MOULDED_FORM_AIM.MAPPED ITEM' IN TYPEOF (it)) |
         NOT
  ('SHIP_MOULDED_FORM_AIM.NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION'
         IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
         = 0;
    WR4: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
         'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
         NOT (SIZEOF (QUERY (cfs <*
         fbsm\face_based_surface_model.fbsm_faces |
         NOT (SIZEOF (['SHIP_MOULDED_FORM_AIM.OPEN_SHELL',
         'SHIP_MOULDED_FORM_AIM.ORIENTED_OPEN_SHELL',
         'SHIP_MOULDED_FORM_AIM.ORIENTED_CLOSED_SHELL',
         'SHIP_MOULDED_FORM_AIM.CLOSED_SHELL']
         * TYPEOF (cfs)) = 1))) = 0))) = 0;
    WR5: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
         'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
         NOT (SIZEOF (QUERY (cfs <*
         fbsm\face_based_surface_model.fbsm_faces |
         NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
         NOT (SIZEOF (['SHIP_MOULDED_FORM_AIM.FACE_SURFACE',
         'SHIP_MOULDED_FORM_AIM.ORIENTED_FACE'] * TYPEOF (fa)) = 1))) =
  0)))
         = 0))) = 0;
    WR6: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
         'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
         NOT (SIZEOF (QUERY (cfs <*
         fbsm\face_based_surface_model.fbsm_faces |
         NOT (SIZEOF (QUERY (f_sf <* QUERY (fa <* cfs.cfs_faces |
```

```
        'SHIP_MOULDED_FORM_AIM.FACE_SURFACE' IN TYPEOF (fa)) |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (f_sf))
        OR
        (SIZEOF (['SHIP_MOULDED_FORM_AIM.B_SPLINE_SURFACE',
        'SHIP_MOULDED_FORM_AIM.ELEMENTARY_SURFACE',
        'SHIP_MOULDED_FORM_AIM.OFFSET_SURFACE',
        'SHIP_MOULDED_FORM_AIM.SURFACE_REPLICA',
        'SHIP_MOULDED_FORM_AIM.SWEPT_SURFACE'] * TYPEOF
        (f_sf\face_surface.face_geometry)) = 1)))) = 0))) = 0))) = 0;
  WR7: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (nmsf_surface_check(fa\face_surface.face_geometry))))) = 0)))
        = 0))) = 0;
  WR8: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (bnds <* fa.bounds |
        NOT (SIZEOF (['SHIP_MOULDED_FORM_AIM.EDGE_LOOP',
        'SHIP_MOULDED_FORM_AIM.VERTEX_LOOP']
        * TYPEOF (bnds.bound)) = 1))) = 0)))) = 0))) = 0))) = 0;
  WR9: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items|
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
        'SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds\path.edge_list |
        NOT ('SHIP_MOULDED_FORM_AIM.EDGE_CURVE' IN TYPEOF
        (oe.edge_element)))) = 0))) = 0)))) = 0))) = 0))) = 0;
 WR10: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
        'SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe_cv <* QUERY (oe <*
        elp_fbnds\path.edge_list |
        'SHIP_MOULDED_FORM_AIM.EDGE_CURVE' IN TYPEOF (oe.edge_element)) |
        NOT (SIZEOF (['SHIP_MOULDED_FORM_AIM.B_SPLINE_CURVE',
        'SHIP_MOULDED_FORM_AIM.CONIC',
        'SHIP_MOULDED_FORM_AIM.CURVE_REPLICA',
        'SHIP_MOULDED_FORM_AIM.LINE',
        'SHIP_MOULDED_FORM_AIM.OFFSET_CURVE_3D',
        'SHIP_MOULDED_FORM_AIM.PCURVE',
        'SHIP_MOULDED_FORM_AIM.POLYLINE',
        'SHIP_MOULDED_FORM_AIM.SURFACE_CURVE'] *
```

```
        TYPEOF (oe_cv.edge_element\edge_curve.edge_geometry))
        = 1))) = 0))) = 0)))) = 0))) = 0))) = 0;
  WR11: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
        'SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds\path.edge_list |
        NOT (nmsf_curve_check
 (oe.edge_element\edge_curve.edge_geometry))))
        = 0))) = 0)))) = 0))) = 0))) = 0;
  WR12: SIZEOF (QUERY(fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
        'SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds\path.edge_list|
        NOT (('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN TYPEOF
        (oe.edge_element.edge_start))
        AND
        ('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN
        TYPEOF (oe.edge_element.edge_end)))))
        = 0))) = 0)))) = 0))) = 0))) = 0;
  WR13: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
        'SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds\path.edge_list |
        NOT ((SIZEOF (['SHIP_MOULDED_FORM_AIM.CARTESIAN_POINT',
        'SHIP_MOULDED_FORM_AIM.DEGENERATE_PCURVE',
        'SHIP_MOULDED_FORM_AIM.POINT_ON_CURVE',
        'SHIP_MOULDED_FORM_AIM.POINT_ON_SURFACE'] * TYPEOF
        (oe.edge_element.edge_start\vertex_point.vertex_geometry)) = 1)
        AND
        (SIZEOF (['SHIP_MOULDED_FORM_AIM.CARTESIAN_POINT',
        'SHIP_MOULDED_FORM_AIM.DEGENERATE_PCURVE',
        'SHIP_MOULDED_FORM_AIM.POINT_ON_CURVE',
        'SHIP_MOULDED_FORM_AIM.POINT_ON_SURFACE'] * TYPEOF
        (oe.edge_element.edge_end\vertex_point.vertex_geometry)) = 1
        )))) = 0))) = 0)))) = 0))) = 0))) = 0;
  WR14: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
        'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cfs <*
        fbsm\face_based_surface_model.fbsm_faces |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
```

```
       'SHIP_MOULDED_FORM_AIM.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
       NOT ('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN TYPEOF
       (vlp_fbnds\vertex_loop.loop_vertex)))) = 0)))) = 0)))
       = 0))) = 0;
 WR15: SIZEOF (QUERY (fbsm <* QUERY (it <* SELF.items |
       'SHIP_MOULDED_FORM_AIM.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (cfs <*
       fbsm\face_based_surface_model.fbsm_faces |
       NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
       NOT (('SHIP_MOULDED_FORM_AIM.ADVANCED_FACE' IN TYPEOF (fa))
       OR
       (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
       'SHIP_MOULDED_FORM_AIM.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (SIZEOF (['SHIP_MOULDED_FORM_AIM.CARTESIAN_POINT',
       'SHIP_MOULDED_FORM_AIM.DEGENERATE_PCURVE',
       'SHIP_MOULDED_FORM_AIM.POINT_ON_CURVE',
       'SHIP_MOULDED_FORM_AIM.POINT_ON_SURFACE'] * TYPEOF
       (vlp_fbnds\vertex_loop.loop_vertex\vertex_point.vertex_geometry))
       = 1))) = 0)))) = 0))) = 0))) = 0;
END_ENTITY;

ENTITY shell_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF.items |
       NOT
(SIZEOF(['SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL',
                    'SHIP_MOULDED_FORM_AIM.MAPPED_ITEM',
                    'SHIP_MOULDED_FORM_AIM.AXIS2_PLACEMENT_3D'] *
            TYPEOF (it)) = 1
       ))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF.items |
             SIZEOF(['SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL',
             'SHIP_MOULDED_FORM_AIM.MAPPED_ITEM'] * TYPEOF (it)) = 1
       )) >= 1;
  WR3: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                ws\wire_shell.wire_shell_extent |
                 ('SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (wsb))) |
              NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
                NOT ('SHIP_MOULDED_FORM_AIM.EDGE_CURVE' IN
                        TYPEOF (el.edge_element)))) = 0)
            )) = 0)
          )) = 0)
       )) = 0;
  WR4: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                ws\wire_shell.wire_shell_extent |
                ('SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (wsb))) |
```

```
              NOT (SIZEOF (QUERY (pline_el <*
                          QUERY (el <* eloop\path.edge_list |
                     ('SHIP_MOULDED_FORM_AIM.POLYLINE' IN
                         TYPEOF
(el.edge_element\edge_curve.edge_geometry))) |
                NOT (SIZEOF (pline_el.edge_element\edge_curve.
                             edge_geometry\polyline.points) >2)
              )) = 0)
            )) = 0)
          )) = 0)
        )) = 0;
  WR5: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
               IN TYPEOF (it))) |
          NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                 ws\wire_shell.wire_shell_extent |
                  ('SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (wsb))) |
                NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
                   NOT (valid_wireframe_edge_curve
                          (el.edge_element\edge_curve.edge_geometry ))))
=0)
           )) = 0)
          )) = 0)
        )) = 0;
  WR6: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
           ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
                IN TYPEOF(it))) |
          NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                 ws\wire_shell.wire_shell_extent |
                    ('SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (wsb))) |
              NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
                NOT (('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN
                         TYPEOF (el.edge_element.edge_start))
                 AND
                     ('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN
                         TYPEOF (el.edge_element.edge_end))) )) = 0)
            )) = 0)
          )) = 0)
        )) = 0;
  WR7: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
               IN TYPEOF (it))) |
          NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                 ws\wire_shell.wire_shell_extent |
                ('SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (wsb))) |
              NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
                     NOT ((valid_wireframe_vertex_point
                            (el.edge_element.
                             edge_start\vertex_point.vertex_geometry))
```

```
                 AND
                   (valid_wireframe_vertex_point

(el.edge_element.edge_end\vertex_point.vertex_geometry)))
             )) = 0)
           )) = 0)
         )) = 0)
       )) = 0;
  WR8: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                   ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                               ws\wire_shell.wire_shell_extent |
               ('SHIP_MOULDED_FORM_AIM.EDGE_LOOP' IN TYPEOF (wsb))) |
             NOT (SIZEOF (QUERY (con_edges <* QUERY (el <*
                                 eloop\path.edge_list |
               ('SHIP_MOULDED_FORM_AIM.CONIC' IN
                   TYPEOF (el.edge_element\edge_curve.edge_geometry))) |
               NOT ('SHIP_MOULDED_FORM_AIM.AXIS2_PLACEMENT_3D' IN
                    TYPEOF (con_edges.edge_element\edge_curve.
                            edge_geometry\conic.position))
             )) = 0)
           )) = 0)
         )) = 0)
       )) = 0;
  WR9: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb)))
|
           NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
                               ws\wire_shell.wire_shell_extent |
                 ('SHIP_MOULDED_FORM_AIM.VERTEX_LOOP' IN TYPEOF (wsb)))
|
             NOT ('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN
                    TYPEOF (vloop\vertex_loop.loop_vertex))
           )) = 0)
         )) = 0)
       )) = 0;
  WR10: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN TYPEOF (sb)))
|
           NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
                               ws\wire_shell.wire_shell_extent |
                 ('SHIP_MOULDED_FORM_AIM.VERTEX_LOOP' IN TYPEOF (wsb)))
|
             NOT (valid_wireframe_vertex_point (vloop\vertex_loop.
                    loop_vertex\vertex_point.vertex_geometry))
           )) = 0)
```

```
          )) = 0)
        )) = 0;
  WR11: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
          NOT (SIZEOF (QUERY (vs <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                  ('SHIP_MOULDED_FORM_AIM.VERTEX_SHELL' IN TYPEOF (sb)))
|
            NOT ('SHIP_MOULDED_FORM_AIM.VERTEX_POINT' IN
                    TYPEOF
(vs\vertex_shell.vertex_shell_extent.loop_vertex))
          )) = 0)
        )) = 0;
  WR12: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('SHIP_MOULDED_FORM_AIM.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
          NOT (SIZEOF (QUERY (vs <* QUERY (sb <*

sbwm\shell_based_wireframe_model.sbwm_boundary |
                  ('SHIP_MOULDED_FORM_AIM.VERTEX_SHELL' IN TYPEOF (sb)))
|
            NOT (valid_wireframe_vertex_point (vs\vertex_shell.
                   vertex_shell_extent.loop_vertex\vertex_point.
                   vertex_geometry))
          )) = 0)
        )) = 0;
  WR13: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
           ('SHIP_MOULDED_FORM_AIM.MAPPED_ITEM' IN TYPEOF (it))) |
         NOT ('SHIP_MOULDED_FORM_AIM.' +

                 'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION' IN
           TYPEOF (mi\mapped_item.mapping_source.mapped_representation)
        ))) = 0;
  WR14: SELF.context_of_items\geometric_representation_context.
             coordinate_space_dimension = 3;
END_ENTITY;

  ENTITY document;
    id                  : identifier;
    name                : label;
    description         : OPTIONAL text;
    kind                : document_type;
  INVERSE
    representation_type : SET[0:?] OF document_representation_type
                            FOR represented_document;
  END_ENTITY; -- document

  ENTITY document_usage_constraint;
    source                : document;
    subject_element       : label;
    subject_element_value : text;
  END_ENTITY; -- document_usage_constraint

  ENTITY group;
    name        : label;
    description : OPTIONAL text;
  END_ENTITY; -- group

  ENTITY group_relationship;
```

```
  name           : label;
  description    : OPTIONAL text;
  relating_group : group;
  related_group  : group;
END_ENTITY; -- group_relationship

ENTITY product;
  id                 : identifier;
  name               : label;
  description        : OPTIONAL text;
  frame_of_reference : SET [1:?] OF product_context;
END_ENTITY; -- product

ENTITY product_definition;
  id                 : identifier;
  description        : OPTIONAL text;
  formation          : product_definition_formation;
  frame_of_reference : product_definition_context;
DERIVE
  name               : label := get_name_value (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- product_definition

ENTITY product_definition_relationship;
  id                          : identifier;
  name                        : label;
  description                 : OPTIONAL text;
  relating_product_definition : product_definition;
  related_product_definition  : product_definition;
END_ENTITY; -- product_definition_relationship

ENTITY product_category;
  name        : label;
  description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
  products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY product_definition_shape
  SUBTYPE OF (property_definition);
UNIQUE
  UR1: SELF\property_definition.definition;
WHERE
  WR1: NOT ('SHIP_MOULDED_FORM_AIM.SHAPE_DEFINITION'
       IN TYPEOF (SELF\property_definition.definition));
END_ENTITY; -- product_definition_shape

ENTITY property_definition;
  name        : label;
  description : OPTIONAL text;
  definition  : characterized_definition;
END_ENTITY; -- property_definition

ENTITY shape_aspect;
  name                  : label;
  description           : OPTIONAL text;
```

```
      of_shape             : product_definition_shape;
      product_definitional : LOGICAL;
    END_ENTITY; -- shape_aspect

    ENTITY organization;
      id          : OPTIONAL identifier;
      name        : label;
      description : OPTIONAL text;
    END_ENTITY; -- organization

    ENTITY organizational_project;
      name                       : label;
      description                : OPTIONAL text;
      responsible_organizations : SET[1:?] OF organization;
    DERIVE
      id               : identifier := get_id_value (SELF);
    WHERE
      WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                            'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
    END_ENTITY; -- organizational_project

  ENTITY property_definition_relationship;
    name                        : label;
    description                  : text;
    relating_property_definition : property_definition;
    related_property_definition  : property_definition;
  END_ENTITY;

    ENTITY property_definition_representation;
      definition          : represented_definition;
      used_representation : representation;
    DERIVE
      description         : text := get_description_value (SELF);
      name                : label := get_name_value (SELF);
    WHERE
      WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                            'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
      WR2 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                            'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
    END_ENTITY; -- property_definition_representation

    ENTITY shape_representation
      SUBTYPE OF (representation);
    END_ENTITY; -- shape_representation

   ENTITY axis2_placement_3d
     SUBTYPE OF (placement);
     axis          : OPTIONAL direction;
     ref_direction : OPTIONAL direction;
   DERIVE
     p             : LIST [3:3] OF direction :=
  build_axes(axis,ref_direction);
   WHERE
     WR1: SELF\placement.location.dim = 3;
     WR2: (NOT (EXISTS (axis))) OR (axis.dim = 3);
     WR3: (NOT (EXISTS (ref_direction))) OR (ref_direction.dim = 3);
     WR4: (NOT (EXISTS (axis))) OR (NOT (EXISTS (ref_direction))) OR
            (cross_product(axis,ref_direction).magnitude > 0.0);
   END_ENTITY;

   ENTITY axis1_placement
```

```
    SUBTYPE OF (placement);
    axis     : OPTIONAL direction;
  DERIVE
    z        : direction := NVL(normalise(axis),
direction([0.0,0.0,1.0]));
  WHERE
    WR1: SELF\geometric_representation_item.dim  = 3;
  END_ENTITY;

  ENTITY cartesian_point
    SUBTYPE OF (point);
    coordinates  : LIST [1:3] OF length_measure;
  END_ENTITY;

  ENTITY elementary_surface
    SUPERTYPE OF (ONEOF (
                PLANE))
   SUBTYPE OF (surface);
    position : axis2_placement_3d;
  END_ENTITY;

  ENTITY placement
    SUPERTYPE OF (ONEOF (
                AXIS1_PLACEMENT,
                AXIS2_PLACEMENT_3D))
   SUBTYPE OF (geometric_representation_item);
    location : cartesian_point;
  END_ENTITY;

  ENTITY surface
    SUPERTYPE OF (ONEOF (
                ELEMENTARY_SURFACE,
                SWEPT_SURFACE,
                BOUNDED_SURFACE,
                OFFSET_SURFACE,
                SURFACE_REPLICA))
   SUBTYPE OF (geometric_representation_item);
  END_ENTITY;

  ENTITY vector
    SUBTYPE OF (geometric_representation_item);
    orientation : direction;
    magnitude   : length_measure;
  WHERE
    WR1 : magnitude >= 0.0;
  END_ENTITY;

  ENTITY point
    SUPERTYPE OF (ONEOF (
                CARTESIAN_POINT,
                POINT_REPLICA))
   SUBTYPE OF (geometric_representation_item);
  END_ENTITY;

  ENTITY curve
    SUPERTYPE OF (ONEOF (
                CONIC,
                PCURVE,
                SURFACE_CURVE,
                OFFSET_CURVE_3D,
                CURVE_REPLICA))
```

```
    SUBTYPE OF (geometric_representation_item);
   END_ENTITY;

   ENTITY plane
   SUBTYPE OF (elementary_surface);
   END_ENTITY;

  ENTITY geometric_representation_item
    SUPERTYPE OF (ONEOF (
                POINT,
                DIRECTION,
                VECTOR,
                PLACEMENT,
                CARTESIAN_TRANSFORMATION_OPERATOR,
                CURVE,
                SURFACE,
                EDGE_CURVE,
                FACE_SURFACE,
                VERTEX_POINT,
                FACE_BASED_SURFACE_MODEL,
                SHELL_BASED_WIREFRAME_MODEL))
    SUBTYPE OF (representation_item);
  DERIVE
    dim : dimension_count := dimension_of(SELF);
  WHERE
    WR1: SIZEOF (QUERY (using_rep <* using_representations (SELF) |
         NOT ('SHIP_MOULDED_FORM_AIM.GEOMETRIC_REPRESENTATION_CONTEXT' IN
         TYPEOF (using_rep.context_of_items)))) = 0;
  END_ENTITY;

  ENTITY geometric_representation_context
    SUBTYPE OF (representation_context);
    coordinate_space_dimension : dimension_count;
  END_ENTITY;

   ENTITY vertex
     SUBTYPE OF (topological_representation_item);
   END_ENTITY;

      ENTITY vertex_point
      SUBTYPE OF(vertex,geometric_representation_item);
        vertex_geometry : point;
      END_ENTITY;

   ENTITY wire_shell
     SUBTYPE OF (topological_representation_item);
     wire_shell_extent : SET [1:?] OF loop;
   WHERE
     WR1: NOT mixed_loop_type_set(wire_shell_extent);
   END_ENTITY;

   ENTITY topological_representation_item
     SUPERTYPE OF (ONEOF (
                VERTEX,
                EDGE,
                FACE_BOUND,
                FACE,
                VERTEX_SHELL,
                WIRE_SHELL,
                CONNECTED_FACE_SET,(
                LOOP
```

```
              ANDOR
              PATH)))
  SUBTYPE OF (representation_item);
 END_ENTITY;

 ENTITY edge_curve
   SUBTYPE OF(edge,geometric_representation_item);
   edge_geometry : curve;
   same_sense    : BOOLEAN;
 END_ENTITY;

 ENTITY shell_based_wireframe_model
   SUBTYPE OF (geometric_representation_item);
   sbwm_boundary  : SET [1:?] OF shell;
 WHERE
   WR1: constraints_geometry_shell_based_wireframe_model(SELF);
 END_ENTITY;

 ENTITY face_based_surface_model
   SUBTYPE OF (geometric_representation_item);
   fbsm_faces : SET [1:?] OF connected_face_set;
 END_ENTITY;

  ENTITY external_source;
    source_id   : source_item;
  DERIVE
    description : text := get_description_value (SELF);
  WHERE
    WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                          'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
  END_ENTITY; -- external_source

  ENTITY external_source_relationship;
    name            : label;
    description     : OPTIONAL text;
    relating_source : external_source;
    related_source  : external_source;
  END_ENTITY; -- external_source_relationship

ENTITY descriptive_representation_item
  SUBTYPE OF (representation_item);
  description : text;
END_ENTITY;

  ENTITY named_unit
    SUPERTYPE OF (ONEOF (
              SI_UNIT,
              CONVERSION_BASED_UNIT));
    dimensions : dimensional_exponents;
  END_ENTITY; -- named_unit

  ENTITY si_unit
    SUBTYPE OF (named_unit);
    prefix     : OPTIONAL si_prefix;
    name       : si_unit_name;
  DERIVE
    SELF\named_unit.dimensions : dimensional_exponents
                               := dimensions_for_si_unit (name);
  END_ENTITY; -- si_unit

  ENTITY conversion_based_unit
```

```
      SUBTYPE OF (named_unit);
      name              : label;
      conversion_factor : measure_with_unit;
    END_ENTITY; -- conversion_based_unit

    ENTITY global_unit_assigned_context
      SUBTYPE OF (representation_context);
      units : SET [1:?] OF unit;
    END_ENTITY; -- global_unit_assigned_context

    ENTITY measure_with_unit
      ;
      value_component : measure_value;
      unit_component  : unit;
    WHERE
      WR1: valid_units (SELF);
    END_ENTITY; -- measure_with_unit

  ENTITY representation;
    name             : label;
    items            : SET[1:?] OF representation_item;
    context_of_items : representation_context;
  DERIVE
    id               : identifier := get_id_value (SELF);
    description      : text := get_description_value (SELF);
  WHERE
    WR1: SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                               'ID_ATTRIBUTE.IDENTIFIED_ITEM'))
         <= 1;
    WR2: SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                               'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM'))
         <= 1;
   END_ENTITY;

  ENTITY representation_relationship;
    name        : label;
    description : OPTIONAL text;
    rep_1       : representation;
    rep_2       : representation;
  END_ENTITY;

  ENTITY representation_context;
    context_identifier : identifier;
    context_type       : text;
  INVERSE
    representations_in_context : SET [1:?] OF representation
      FOR context_of_items;
  END_ENTITY;

  ENTITY representation_item;
    name : label;
  WHERE
    WR1: SIZEOF(using_representations(SELF)) > 0;
  END_ENTITY;

  ENTITY value_representation_item
  SUBTYPE OF (representation_item);
    value_component : measure_value;
  WHERE
    WR1: SIZEOF (QUERY (rep <* using_representations (SELF) |
         NOT ('SHIP_MOULDED_FORM_AIM.GLOBAL_UNIT_ASSIGNED_CONTEXT'
```

```
      IN TYPEOF (rep.context_of_items)
      ))) = 0;
END_ENTITY;

ENTITY compound_representation_item
  SUBTYPE OF (representation_item);
  item_element : compound_item_definition;
END_ENTITY;

ENTITY item_defined_transformation;
  name             : label;
  description      : OPTIONAL text;
  transform_item_1 : representation_item;
  transform_item_2 : representation_item;
END_ENTITY;

  ENTITY id_attribute;
    attribute_value : identifier;
    identified_item : id_attribute_select;
  END_ENTITY; -- id_attribute

  ENTITY description_attribute;
    attribute_value : text;
    described_item  : description_attribute_select;
  END_ENTITY; -- description_attribute

  ENTITY name_attribute;
    attribute_value : label;
    named_item      : name_attribute_select;
  END_ENTITY; -- name_attribute

  ENTITY classification_role;
    name        : label;
    description : OPTIONAL text;
  END_ENTITY; -- classification_role

  ENTITY object_role;
    name        : label;
    description : OPTIONAL text;
  END_ENTITY; -- object_role

  ENTITY role_association;
    role           : object_role;
    item_with_role : role_select;
  END_ENTITY; -- role_association

  ENTITY identification_role;
    name        : label;
    description : OPTIONAL text;
  END_ENTITY; -- identification_role

  ENTITY organization_role;
    name        : label;
  DERIVE
    description : text := get_description_value (SELF);
  WHERE
    WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                          'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
  END_ENTITY; -- organization_role

  ENTITY date_and_time;
```

```
  date_component : date;
  time_component : local_time;
END_ENTITY; -- date_and_time

ENTITY date
  ;
  year_component : year_number;
END_ENTITY; -- date

ENTITY local_time;
  hour_component   : hour_in_day;
  minute_component : OPTIONAL minute_in_hour;
  second_component : OPTIONAL second_in_minute;
  zone             : coordinated_universal_time_offset;
WHERE
  WR1: valid_time (SELF);
END_ENTITY; -- local_time

ENTITY coordinated_universal_time_offset;
  hour_offset   : INTEGER;
  minute_offset : OPTIONAL INTEGER;
  sense         : ahead_or_behind;
WHERE
  WR1: { 0 <= hour_offset <= 12 };
  WR2: { 0 <= minute_offset <= 59 };
  WR3: NOT ((hour_offset <> 0) AND (sense = exact));
END_ENTITY; -- coordinated_universal_time_offset

ENTITY date_time_role;
  name        : label;
DERIVE
  description : text := get_description_value (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- date_time_role

ENTITY person_and_organization;
  the_person       : person;
  the_organization : organization;
DERIVE
  name             : label := get_name_value (SELF);
  description      : text := get_description_value (SELF);
WHERE
  WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
  WR2 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                        'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- person_and_organization

ENTITY person;
  id            : identifier;
  last_name     : OPTIONAL label;
  first_name    : OPTIONAL label;
  middle_names  : OPTIONAL LIST [1:?] OF label;
  prefix_titles : OPTIONAL LIST [1:?] OF label;
  suffix_titles : OPTIONAL LIST [1:?] OF label;
WHERE
  WR1: EXISTS(last_name) OR EXISTS(first_name);
END_ENTITY; -- person
```

```
  ENTITY person_and_organization_role;
    name        : label;
  DERIVE
    description : text := get_description_value (SELF);
  WHERE
    WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +

                          'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
  END_ENTITY; -- person_and_organization_role

ENTITY mapped_item
  SUBTYPE OF (representation_item);
  mapping_source : representation_map;
  mapping_target : representation_item;
WHERE
  WR1: acyclic_mapped_representation(using_representations(SELF),
[SELF]);
END_ENTITY;

ENTITY representation_map;
  mapping_origin        : representation_item;
  mapped_representation : representation;
INVERSE
  map_usage : SET[1:?] OF mapped_item FOR mapping_source;
WHERE
  WR1: item_in_context(SELF.mapping_origin,
       SELF.mapped_representation.context_of_items);
END_ENTITY;

 ENTITY face_surface
   SUBTYPE OF(face,geometric_representation_item);
   face_geometry :  surface;
   same_sense    :  BOOLEAN;
 END_ENTITY;

 ENTITY face
   SUPERTYPE OF (ONEOF (
               FACE_SURFACE))
  SUBTYPE OF (topological_representation_item);
   bounds : SET[1:?] OF face_bound;
 WHERE
   WR1: NOT (mixed_loop_type_set(list_to_set(list_face_loops(SELF))));
   WR2: SIZEOF(QUERY(temp <* bounds |
'SHIP_MOULDED_FORM_AIM.FACE_OUTER_BOUND'
         IN TYPEOF(temp))) <= 1;
 END_ENTITY;

 ENTITY face_bound
   SUBTYPE OF(topological_representation_item);
   bound       :  loop;
   orientation :  BOOLEAN;
 END_ENTITY;

 ENTITY loop
   SUPERTYPE OF (ONEOF (
               VERTEX_LOOP))
  SUBTYPE OF (topological_representation_item);
 END_ENTITY;

 ENTITY b_spline_curve
```

```
    SUBTYPE OF (bounded_curve);
     degree                : INTEGER;
     control_points_list   : LIST [2:?] OF cartesian_point;
     curve_form            : b_spline_curve_form;
     closed_curve          : LOGICAL;
     self_intersect        : LOGICAL;
   DERIVE
     upper_index_on_control_points  : INTEGER
                                    := (SIZEOF(control_points_list) - 1);
     control_points        : ARRAY [0:upper_index_on_control_points]
                                                          OF
  cartesian_point
                                    := list_to_array(control_points_list,0,

  upper_index_on_control_points);
   WHERE
     WR1: ('SHIP_MOULDED_FORM_AIM.UNIFORM_CURVE' IN TYPEOF(self)) OR
          ('SHIP_MOULDED_FORM_AIM.QUASI_UNIFORM_CURVE' IN TYPEOF(self)) OR
          ('SHIP_MOULDED_FORM_AIM.BEZIER_CURVE' IN TYPEOF(self)) OR
          ('SHIP_MOULDED_FORM_AIM.B_SPLINE_CURVE_WITH_KNOTS' IN
  TYPEOF(self));

   END_ENTITY;

   ENTITY bounded_curve
     SUPERTYPE OF (ONEOF (
                POLYLINE,
                B_SPLINE_CURVE,
                COMPOSITE_CURVE))
    SUBTYPE OF (curve);
   END_ENTITY;

   ENTITY curve_replica
     SUBTYPE OF (curve);
     parent_curve   : curve;
     transformation : cartesian_transformation_operator;
   WHERE
     WR1: transformation.dim = parent_curve.dim;
     WR2: acyclic_curve_replica (SELF, parent_curve);
   END_ENTITY;

   ENTITY cartesian_transformation_operator
     SUPERTYPE OF (ONEOF (
                CARTESIAN_TRANSFORMATION_OPERATOR_3D))
    SUBTYPE OF (geometric_representation_item,
                              functionally_defined_transformation);
     axis1        : OPTIONAL direction;
     axis2        : OPTIONAL direction;
     local_origin : cartesian_point;
     scale        : OPTIONAL REAL;
   DERIVE
     scl          : REAL := NVL(scale, 1.0);
   WHERE
     WR1: scl > 0.0;
   END_ENTITY;

  ENTITY functionally_defined_transformation;
    name        : label;
    description : OPTIONAL text;
  END_ENTITY;
```

```
 ENTITY direction
   SUBTYPE OF (geometric_representation_item);
   direction_ratios : LIST [2:3] OF REAL;
 WHERE
   WR1: SIZEOF(QUERY(tmp <* direction_ratios | tmp <> 0.0)) > 0;
 END_ENTITY;

 ENTITY offset_curve_3d
   SUBTYPE OF (curve);
   basis_curve    : curve;
   distance       : length_measure;
   self_intersect : LOGICAL;
   ref_direction  : direction;
 WHERE
   WR1 : (basis_curve.dim = 3) AND (ref_direction.dim = 3);
 END_ENTITY;

 ENTITY pcurve
   SUBTYPE OF (curve);
   basis_surface      : surface;
   reference_to_curve : definitional_representation;
 WHERE
   WR1: SIZEOF(reference_to_curve\representation.items) = 1;
   WR2: 'SHIP_MOULDED_FORM_AIM.CURVE' IN TYPEOF

                     (reference_to_curve\representation.items[1]);
   WR3: reference_to_curve\representation.
                    items[1]\geometric_representation_item.dim = 2;
 END_ENTITY;

ENTITY definitional_representation
  SUBTYPE OF ( representation );
WHERE
  WR1: 'SHIP_MOULDED_FORM_AIM.PARAMETRIC_REPRESENTATION_CONTEXT' IN

        TYPEOF( SELF\representation.context_of_items );
END_ENTITY;

 ENTITY surface_curve

  SUBTYPE OF (curve);
   curve_3d              : curve;
   associated_geometry   : LIST[1:2] OF pcurve_or_surface;
   master_representation : preferred_surface_curve_representation;
 DERIVE
   basis_surface         : SET[1:2] OF surface
                         := get_basis_surface(SELF);
 WHERE
   WR1: curve_3d.dim = 3;
   WR2: ('SHIP_MOULDED_FORM_AIM.PCURVE' IN
TYPEOF(associated_geometry[1])) OR
                        (master_representation <> pcurve_s1);
   WR3: ('SHIP_MOULDED_FORM_AIM.PCURVE' IN
TYPEOF(associated_geometry[2])) OR
                        (master_representation <> pcurve_s2);
   WR4: NOT ('SHIP_MOULDED_FORM_AIM.PCURVE' IN TYPEOF(curve_3d));

 END_ENTITY;

 ENTITY composite_curve_on_surface
```

```
 SUBTYPE OF (composite_curve);

DERIVE
  basis_surface : SET[0:2] OF surface :=
                 get_basis_surface(SELF);
WHERE
  WR1: SIZEOF(basis_surface) > 0;
  WR2: constraints_composite_curve_on_surface(SELF);
END_ENTITY;

ENTITY composite_curve
  SUBTYPE OF (bounded_curve);
  segments       : LIST [1:?] OF composite_curve_segment;
  self_intersect : LOGICAL;
DERIVE
  n_segments     : INTEGER := SIZEOF(segments);
  closed_curve   : LOGICAL
                 := segments[n_segments].transition <> discontinuous;
WHERE
  WR1: ((NOT closed_curve) AND (SIZEOF(QUERY(temp <* segments |
                 temp.transition = discontinuous)) = 1)) OR
            ((closed_curve) AND (SIZEOF(QUERY(temp <* segments |
                 temp.transition = discontinuous)) = 0));
END_ENTITY;

ENTITY composite_curve_segment;
  transition    : transition_code;
  same_sense    : BOOLEAN;
  parent_curve  : curve;
INVERSE
  using_curves  : BAG[1:?] OF composite_curve FOR segments;
WHERE
  WR1 : ('SHIP_MOULDED_FORM_AIM.BOUNDED_CURVE' IN TYPEOF(parent_curve));

END_ENTITY;

ENTITY polyline
  SUBTYPE OF (bounded_curve);
  points : LIST [2:?] OF cartesian_point;
END_ENTITY;

ENTITY swept_surface

 SUBTYPE OF (surface);
  swept_curve : curve;
END_ENTITY;

ENTITY offset_surface
  SUBTYPE OF (surface);
  basis_surface  : surface;
  distance       : length_measure;
  self_intersect : LOGICAL;
END_ENTITY;

ENTITY surface_replica
  SUBTYPE OF (surface);
  parent_surface : surface;
  transformation : cartesian_transformation_operator_3d;
WHERE
  WR1: acyclic_surface_replica(SELF, parent_surface);
END_ENTITY;
```

```
 ENTITY cartesian_transformation_operator_3d
   SUBTYPE OF (cartesian_transformation_operator);
   axis3 : OPTIONAL direction;
 DERIVE
   u     : LIST[3:3] OF direction
         := base_axis(3,SELF\cartesian_transformation_operator.axis1,

SELF\cartesian_transformation_operator.axis2,axis3);
 WHERE
   WR1: SELF\cartesian_transformation_operator.dim = 3;
 END_ENTITY;

 ENTITY b_spline_surface

  SUBTYPE OF (bounded_surface);
   u_degree             : INTEGER;
   v_degree             : INTEGER;
   control_points_list  : LIST [2:?] OF
                            LIST [2:?] OF cartesian_point;
   surface_form         : b_spline_surface_form;
   u_closed             : LOGICAL;
   v_closed             : LOGICAL;
   self_intersect       : LOGICAL;
 DERIVE
   u_upper             : INTEGER := SIZEOF(control_points_list) - 1;
   v_upper             : INTEGER := SIZEOF(control_points_list[1]) - 1;
   control_points      : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
                         cartesian_point
                       := make_array_of_array(control_points_list,
                                              0,u_upper,0,v_upper);
 WHERE
   WR1: ('SHIP_MOULDED_FORM_AIM.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
        ('SHIP_MOULDED_FORM_AIM.QUASI_UNIFORM_SURFACE' IN TYPEOF(SELF))
OR
        ('SHIP_MOULDED_FORM_AIM.BEZIER_SURFACE' IN TYPEOF(SELF)) OR
        ('SHIP_MOULDED_FORM_AIM.B_SPLINE_SURFACE_WITH_KNOTS' IN
TYPEOF(SELF));

 END_ENTITY;

 ENTITY bounded_surface
   SUPERTYPE OF (ONEOF (
               B_SPLINE_SURFACE))
  SUBTYPE OF (surface);
 END_ENTITY;

 ENTITY path

  SUBTYPE OF (topological_representation_item);
   edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
 WHERE
   WR1: path_head_to_tail(SELF);
 END_ENTITY;

 ENTITY oriented_edge
   SUBTYPE OF (edge);
   edge_element : edge;
   orientation  : BOOLEAN;
 DERIVE
   SELF\edge.edge_start : vertex := boolean_choose (SELF.orientation,
```

```
                                                SELF.edge_element.edge_start,
                                                SELF.edge_element.edge_end);
      SELF\edge.edge_end   : vertex := boolean_choose (SELF.orientation,
                                                SELF.edge_element.edge_end,

   SELF.edge_element.edge_start);
    WHERE
      WR1: NOT ('SHIP_MOULDED_FORM_AIM.ORIENTED_EDGE' IN TYPEOF
   (SELF.edge_element));

    END_ENTITY;

    ENTITY edge
      SUPERTYPE OF (ONEOF (
                  EDGE_CURVE,
                  ORIENTED_EDGE))
     SUBTYPE OF (topological_representation_item);
      edge_start : vertex;
      edge_end   : vertex;
    END_ENTITY;

    ENTITY vertex_loop
      SUBTYPE OF (loop);
      loop_vertex : vertex;
    END_ENTITY;

    ENTITY point_replica
      SUBTYPE OF (point);
      parent_pt      : point;
      transformation : cartesian_transformation_operator;
    WHERE
      WR1: transformation.dim = parent_pt.dim;
      WR2: acyclic_point_replica (SELF,parent_pt);
    END_ENTITY;

    ENTITY conic

     SUBTYPE OF (curve);
      position: axis2_placement;
    END_ENTITY;

    ENTITY vertex_shell
      SUBTYPE OF (topological_representation_item);
      vertex_shell_extent : vertex_loop;
    END_ENTITY;

     ENTITY document_type;
       product_data_type : label;
     END_ENTITY; -- document_type

   ENTITY document_representation_type;
       name                 : label;
       represented_document : document;
     END_ENTITY; -- document_representation_type

     ENTITY product_context
       SUBTYPE OF (application_context_element);
       discipline_type : label;
     END_ENTITY; -- product_context

     ENTITY application_context_element
```

```
    SUPERTYPE OF (ONEOF (
              PRODUCT_CONTEXT,
              PRODUCT_DEFINITION_CONTEXT));
    name                : label;
    frame_of_reference : application_context;
  END_ENTITY; -- application_context_element

  ENTITY application_context;
    application      : label;
  DERIVE
    description      : text := get_description_value (SELF);
    id               : identifier := get_id_value (SELF);
  INVERSE
    context_elements : SET [1:?] OF application_context_element
                                FOR frame_of_reference;
  WHERE
    WR1 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +
                 'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
    WR2 : SIZEOF (USEDIN (SELF, 'SHIP_MOULDED_FORM_AIM.' +

                 'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
  END_ENTITY; -- application_context

  ENTITY product_definition_formation;
    id          : identifier;
    description : OPTIONAL text;
    of_product  : product;
  UNIQUE
    UR1: id, of_product;
  END_ENTITY; -- product_definition_formation

  ENTITY product_definition_context
    SUBTYPE OF (application_context_element);
    life_cycle_stage : label;
  END_ENTITY; -- product_definition_context

 ENTITY connected_face_set

  SUBTYPE OF (topological_representation_item);
   cfs_faces : SET [1:?] OF face;
 END_ENTITY;

  ENTITY dimensional_exponents;
    length_exponent                    : REAL;
    mass_exponent                      : REAL;
    time_exponent                      : REAL;
    electric_current_exponent          : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent       : REAL;
    luminous_intensity_exponent        : REAL;
  END_ENTITY; -- dimensional_exponents

FUNCTION get_extended_property_definition_id_value
(obj : extended_property_definition) : identifier;
LOCAL
c_a_set: SET OF applied_identification_assignment := [];
t1_set:  SET OF extended_property_definition:= [];
global_id : identifier;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
```

```
i.ROLE.NAME = 'global id');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
IF obj :=: c_a_set[i].items[j] THEN
    global_id := c_a_set[i].assigned_id;
END_IF;
END_REPEAT;
END_REPEAT;
IF obj IN t1_set
THEN RETURN (global_id);
ELSE RETURN (?);
END_IF;
END_FUNCTION;

FUNCTION get_extended_product_definition_shape_id_value
(obj : extended_product_definition_shape) : identifier;
LOCAL
c_a_set: SET OF applied_identification_assignment := [];
t1_set:  SET OF extended_product_definition_shape := [];
global_id : identifier;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
i.ROLE.NAME = 'global id');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
IF obj :=: c_a_set[i].items[j] THEN
    global_id := c_a_set[i].assigned_id;
END_IF;
END_REPEAT;
END_REPEAT;
IF obj IN t1_set
THEN RETURN (global_id);
ELSE RETURN (?);
END_IF;
END_FUNCTION;

FUNCTION get_extended_product_related_product_category_id_value
(obj : extended_product_related_product_category) : identifier;
LOCAL
c_a_set: SET OF applied_identification_assignment := [];
t1_set:  SET OF extended_product_related_product_category := [];
global_id : identifier;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
i.ROLE.NAME = 'global id');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
IF obj :=: c_a_set[i].items[j] THEN
    global_id := c_a_set[i].assigned_id;
END_IF;
END_REPEAT;
END_REPEAT;
IF obj IN t1_set
```

```
THEN RETURN (global_id);
ELSE RETURN (?);
END_IF;
END_FUNCTION;

  FUNCTION get_id_value
    (obj : id_attribute_select) : identifier;
    LOCAL
      id_bag : BAG OF id_attribute :=
               (USEDIN (obj,
               'SHIP_MOULDED_FORM_AIM.' +

               'ID_ATTRIBUTE.' +
               'IDENTIFIED_ITEM'));
    END_LOCAL;
    IF SIZEOF (id_bag) = 1
      THEN RETURN (id_bag[1].attribute_value);
      ELSE RETURN (?);
    END_IF;
  END_FUNCTION; -- get_id_value

  FUNCTION get_description_value
    (obj : description_attribute_select) : text;
    LOCAL
      description_bag : BAG OF description_attribute :=
                        (USEDIN (obj,
                        'SHIP_MOULDED_FORM_AIM.' +

                        'DESCRIPTION_ATTRIBUTE.' +
                        'DESCRIBED_ITEM'));
    END_LOCAL;
    IF SIZEOF (description_bag) = 1
      THEN RETURN (description_bag[1].attribute_value);
      ELSE RETURN (?);
    END_IF;
  END_FUNCTION; -- get_description_value

  FUNCTION get_name_value
    (obj : name_attribute_select) : label;
    LOCAL
      name_bag : BAG OF name_attribute :=
                 (USEDIN (obj,
                 'SHIP_MOULDED_FORM_AIM.' +

                 'NAME_ATTRIBUTE.' +
                 'NAMED_ITEM'));
    END_LOCAL;
    IF SIZEOF (name_bag) = 1
      THEN RETURN (name_bag[1].attribute_value);
      ELSE RETURN (?);
    END_IF;
  END_FUNCTION; -- get_name_value

  FUNCTION get_role
    (obj : role_select) : object_role;
    LOCAL
    role_bag : BAG OF role_association:=
               (USEDIN (obj,
               'SHIP_MOULDED_FORM_AIM.' +

               'ROLE_ASSOCIATION.' +
```

```
                 'ITEM_WITH_ROLE'));
      END_LOCAL;
      IF SIZEOF (role_bag) = 1
        THEN RETURN (role_bag[1].role);
        ELSE RETURN (?);
      END_IF;
    END_FUNCTION; -- get_role

    FUNCTION valid_time (time: local_time) : BOOLEAN;
      IF EXISTS (time.second_component) THEN
        RETURN (EXISTS (time.minute_component));
      ELSE
        RETURN (TRUE);
      END_IF;
    END_FUNCTION; -- valid_time

  FUNCTION item_in_context
    (item  : representation_item;
     cntxt : representation_context) : BOOLEAN;
    LOCAL
      y : BAG OF representation_item;
    END_LOCAL;

    -- If there is one or more representation using both the item
    -- and cntxt return true.
    IF SIZEOF(USEDIN(item,'SHIP_MOULDED_FORM_AIM.REPRESENTATION.ITEMS')
      * cntxt.representations_in_context) > 0 THEN
      RETURN (TRUE);
      -- Determine the bag of representation_items that reference
      -- item.
      ELSE
      y := QUERY(z <* USEDIN (item , '') |
           'SHIP_MOULDED_FORM_AIM.REPRESENTATION_ITEM' IN TYPEOF(z));

        -- Ensure that the set is not empty.
        IF SIZEOF(y) > 0 THEN
        -- For each element in the set
        REPEAT i := 1 TO HIINDEX(y);
          -- Check to see it is an item in the input cntxt.
          IF item_in_context(y[i], cntxt) THEN
            RETURN (TRUE);
          END_IF;
        END_REPEAT;
      END_IF;
    END_IF;
    -- Return false when all possible branches have been checked
    -- with no success.
    RETURN (FALSE);
  END_FUNCTION;

  FUNCTION acyclic_mapped_representation
    (parent_set   : SET OF representation;
     children_set : SET OF representation_item) : BOOLEAN;
    LOCAL
      x,y : SET OF representation_item;
      i   : INTEGER;
    END_LOCAL;
    -- Determine the subset of children_set that are mapped_items.
    x := QUERY(z <* children_set | 'SHIP_MOULDED_FORM_AIM.MAPPED_ITEM'
         IN TYPEOF(z));
    -- Determine that the subset has elements.
```

```
  IF SIZEOF(x) > 0 THEN
    -- Check each element of the set.
    REPEAT i := 1 TO HIINDEX(x);
      -- If the selected element maps a representation in the
      -- parent_set, then return false.
      IF x[i]\mapped_item.mapping_source.mapped_representation
        IN parent_set THEN
        RETURN (FALSE);
      END_IF;
      -- Recursive check of the items of mapped_rep.
      IF NOT acyclic_mapped_representation
        (parent_set +
        x[i]\mapped_item.mapping_source.mapped_representation,
        x[i]\mapped_item.mapping_source.mapped_representation.items) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  -- Determine the subset of children_set that are not
  -- mapped_items.
  x := children_set - x;
  -- Determine that the subset has elements.
  IF SIZEOF(x) > 0 THEN
    -- For each element of the set:
    REPEAT i := 1 TO HIINDEX(x);
      -- Determine the set of representation_items referenced.
      y := QUERY(z <* bag_to_set( USEDIN(x[i], '')) |
           'SHIP_MOULDED_FORM_AIM.REPRESENTATION_ITEM' IN TYPEOF(z));

      -- Recursively check for an offending mapped_item.
      -- Return false for any errors encountered.
      IF NOT acyclic_mapped_representation(parent_set, y) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  -- Return true when all elements are checked and
  -- no error conditions found.
  RETURN (TRUE);
END_FUNCTION;

  FUNCTION bag_to_set
    (the_bag : BAG OF GENERIC : intype) : SET OF GENERIC : intype;
    LOCAL
      the_set: SET OF GENERIC : intype := [];
    END_LOCAL;
    IF SIZEOF (the_bag) > 0 THEN
      REPEAT i := 1 to HIINDEX (the_bag);
        the_set := the_set + the_bag [i];
      END_REPEAT;
    END_IF;
    RETURN (the_set);
  END_FUNCTION; -- bag_to_set

FUNCTION using_representations (item : representation_item)
  : SET OF representation;
  LOCAL
    results            : SET OF representation;
    result_bag         : BAG OF representation;
    intermediate_items : SET OF representation_item;
  END_LOCAL;
```

```
    -- Find the representations in which the item is used and add to the
    -- results set.
    results := [];
    result_bag :=
  USEDIN(item,'SHIP_MOULDED_FORM_AIM.REPRESENTATION.ITEMS');
    IF SIZEOF(result_bag) > 0 THEN
      REPEAT i := 1 TO HIINDEX(result_bag);
        results := results + result_bag[i];
      END_REPEAT;
    END_IF;
    -- Find the set of representation_items in which item is used.
    intermediate_items := QUERY(z <* bag_to_set( USEDIN(item , '')) |
      'SHIP_MOULDED_FORM_AIM.REPRESENTATION_ITEM' IN TYPEOF(z));

    -- If the set of intermediate items is not empty;
    IF SIZEOF(intermediate_items) > 0 THEN
      -- For each element in the set, recursively add the
      -- using_representations of that element.
      REPEAT i := 1 TO HIINDEX(intermediate_items);
        results := results + using_representations(intermediate_items[i]);
      END_REPEAT;
    END_IF;
    -- Return the set of representation in which the input item is
    -- used directly and indirectly (through intervening
    -- representation_items).
    RETURN (results);
  END_FUNCTION;

    FUNCTION mixed_loop_type_set(l: SET[0:?] OF loop): LOGICAL;
      LOCAL
        i             : INTEGER;
        poly_loop_type: LOGICAL;
      END_LOCAL;
      IF(SIZEOF(l) <= 1) THEN
        RETURN(FALSE);
      END_IF;
      poly_loop_type := ('SHIP_MOULDED_FORM_AIM.POLY_LOOP' IN
  TYPEOF(l[1]));
      REPEAT i := 2 TO SIZEOF(l);
        IF(('SHIP_MOULDED_FORM_AIM.POLY_LOOP' IN TYPEOF(l[i])) <>
  poly_loop_type) THEN

          RETURN(TRUE);
         END_IF;
      END_REPEAT;
      RETURN(FALSE);
    END_FUNCTION;

   FUNCTION list_to_set(l : LIST [0:?] OF GENERIC:T) : SET OF GENERIC:T;
     LOCAL
       s : SET OF GENERIC:T := [];
     END_LOCAL;

     REPEAT i := 1 TO SIZEOF(l);
       s := s + l[i];
     END_REPEAT;

     RETURN(s);
   END_FUNCTION;

   FUNCTION list_face_loops(f: face) : LIST[0:?] OF loop;
```

```
   LOCAL
     loops : LIST[0:?] OF loop := [];
   END_LOCAL;

   REPEAT i := 1 TO SIZEOF(f.bounds);
     loops := loops +(f.bounds[i].bound);
   END_REPEAT;

   RETURN(loops);
 END_FUNCTION;

FUNCTION nmsf_surface_check (surf : surface) : BOOLEAN;

  IF 'SHIP_MOULDED_FORM_AIM.ELEMENTARY_SURFACE' IN TYPEOF(surf) THEN
    RETURN(TRUE);
  ELSE


    IF 'SHIP_MOULDED_FORM_AIM.SWEPT_SURFACE' IN TYPEOF (surf) THEN
      RETURN (nmsf_curve_check(surf\swept_surface.swept_curve));
    ELSE


      IF (('SHIP_MOULDED_FORM_AIM.OFFSET_SURFACE' IN TYPEOF (surf)) AND
        (surf\offset_surface.self_intersect = FALSE)) THEN
        RETURN (nmsf_surface_check(surf\offset_surface.basis_surface));
      ELSE


        IF 'SHIP_MOULDED_FORM_AIM.SURFACE_REPLICA' IN TYPEOF(surf) THEN

RETURN(nmsf_surface_check(surf\surface_replica.parent_surface));
        ELSE


          IF (('SHIP_MOULDED_FORM_AIM.B_SPLINE_SURFACE' IN TYPEOF(surf))
AND

            (surf\b_spline_surface.self_intersect = FALSE)) THEN
            RETURN(TRUE);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;

FUNCTION nmsf_curve_check (cv: curve) : BOOLEAN;

 IF SIZEOF (['SHIP_MOULDED_FORM_AIM.BOUNDED_CURVE',
   'SHIP_MOULDED_FORM_AIM.CONIC',
   'SHIP_MOULDED_FORM_AIM.CURVE_REPLICA', 'SHIP_MOULDED_FORM_AIM.LINE',
   'SHIP_MOULDED_FORM_AIM.OFFSET_CURVE_3D'] * TYPEOF(cv)) > 1
 THEN RETURN(FALSE);
 ELSE


  IF (('SHIP_MOULDED_FORM_AIM.B_SPLINE_CURVE' IN TYPEOF (cv)) AND
    (cv\b_spline_curve.self_intersect = FALSE))
  THEN RETURN(TRUE);
```

```
    ELSE


      IF SIZEOF (['SHIP_MOULDED_FORM_AIM.CONIC',
  'SHIP_MOULDED_FORM_AIM.LINE']
        * TYPEOF (cv)) = 1 THEN
        RETURN(TRUE);
      ELSE


        IF 'SHIP_MOULDED_FORM_AIM.CURVE_REPLICA' IN TYPEOF(cv) THEN
          RETURN (nmsf_curve_check(cv\curve_replica.parent_curve));
        ELSE


          IF (('SHIP_MOULDED_FORM_AIM.OFFSET_CURVE_3D' IN TYPEOF (cv)) AND
            (cv\offset_curve_3d.self_intersect = FALSE)) THEN
            RETURN (nmsf_curve_check(cv\offset_curve_3d.basis_curve));
          ELSE


            IF 'SHIP_MOULDED_FORM_AIM.PCURVE' IN TYPEOF(cv) THEN
              RETURN ((nmsf_curve_check
              (cv\pcurve.reference_to_curve\representation.items[1])) AND
              (nmsf_surface_check(cv\pcurve.basis_surface)));
            ELSE


              IF 'SHIP_MOULDED_FORM_AIM.SURFACE_CURVE' IN TYPEOF(cv) THEN


                IF nmsf_curve_check(cv\surface_curve.curve_3d) THEN
                  REPEAT i := 1 TO SIZEOF
                  (cv\surface_curve.associated_geometry);


                    IF 'SHIP_MOULDED_FORM_AIM.SURFACE' IN
                      TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
                      IF NOT nmsf_surface_check
                        (cv\surface_curve.associated_geometry[i]) THEN
                        RETURN(FALSE);
                      END_IF;
                    ELSE
                      IF 'SHIP_MOULDED_FORM_AIM.PCURVE' IN TYPEOF
                        (cv\surface_curve.associated_geometry[i]) THEN
                        IF NOT nmsf_curve_check
                          (cv\surface_curve.associated_geometry[i]) THEN
                          RETURN(FALSE);
                        END_IF;
                      END_IF;
                    END_IF;
                  END_REPEAT;
                  RETURN(TRUE);
                END_IF;
              ELSE


                IF 'SHIP_MOULDED_FORM_AIM.POLYLINE' IN TYPEOF(cv) THEN

                  IF (SIZEOF (cv\polyline.points) >= 3) THEN RETURN (TRUE);
                  END_IF;
```

```
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
 END_IF;
 RETURN (FALSE);
END_FUNCTION;

 FUNCTION list_to_array(lis : LIST [0:?] OF GENERIC : T;
                      low,u : INTEGER) : ARRAY[low:u] OF GENERIC : T;
   LOCAL
     n   : INTEGER;
     res : ARRAY [low:u] OF GENERIC : T;
   END_LOCAL;

   n := SIZEOF(lis);
   IF (n <> (u-low +1)) THEN
     RETURN(?);
   ELSE
     REPEAT i := 1 TO n;
       res[low+i-1] := lis[i];
     END_REPEAT;
     RETURN(res);
   END_IF;
 END_FUNCTION;

 FUNCTION acyclic_curve_replica(rep : curve_replica; parent : curve)
                                                : BOOLEAN;
   IF NOT (('SHIP_MOULDED_FORM_AIM.CURVE_REPLICA') IN TYPEOF(parent))
THEN

      RETURN (TRUE);
   END_IF;

   IF (parent :=: rep) THEN
      RETURN (FALSE);

    ELSE RETURN(acyclic_curve_replica(rep,
parent\curve_replica.parent_curve));
    END_IF;
  END_FUNCTION;

FUNCTION get_basis_surface (c : curve_on_surface) : SET[0:2] OF surface;
  LOCAL
    surfs  : SET[0:2] OF surface;
    n      : INTEGER;
  END_LOCAL;
  surfs := [];
  IF 'SHIP_MOULDED_FORM_AIM.PCURVE' IN TYPEOF (c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'SHIP_MOULDED_FORM_AIM.SURFACE_CURVE' IN TYPEOF (c) THEN
      n := SIZEOF(c\surface_curve.associated_geometry);
      REPEAT i := 1 TO n;
      surfs := surfs +

associated_surface(c\surface_curve.associated_geometry[i]);
      END_REPEAT;
```

```
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF (c)
  THEN


      n := SIZEOF(c\composite_curve_on_surface.segments);
      surfs :=
  get_basis_surface(c\composite_curve_on_surface.segments[1].parent_curve);
      IF n > 1 THEN
        REPEAT i := 2 TO n;
          surfs := surfs *
  get_basis_surface(c\composite_curve_on_surface.segments[i].parent_curve);
        END_REPEAT;
      END_IF;
    END_IF;
    RETURN(surfs);
  END_FUNCTION;

   FUNCTION associated_surface(arg : pcurve_or_surface) : surface;
     LOCAL
       surf : surface;
     END_LOCAL;

     IF 'SHIP_MOULDED_FORM_AIM.PCURVE' IN TYPEOF(arg) THEN

       surf := arg.basis_surface;
     ELSE
       surf := arg;
     END_IF;
     RETURN(surf);
   END_FUNCTION;

   FUNCTION constraints_composite_curve_on_surface
                 (c: composite_curve_on_surface) : BOOLEAN;
     LOCAL
       n_segments : INTEGER := SIZEOF(c.segments);
     END_LOCAL;

     REPEAT k := 1 TO n_segments;
       IF (NOT('SHIP_MOULDED_FORM_AIM.PCURVE' IN
             TYPEOF(c\composite_curve.segments[k].parent_curve))) AND
          (NOT('SHIP_MOULDED_FORM_AIM.SURFACE_CURVE' IN
             TYPEOF(c\composite_curve.segments[k].parent_curve))) AND
          (NOT('SHIP_MOULDED_FORM_AIM.COMPOSITE_CURVE_ON_SURFACE' IN

             TYPEOF(c\composite_curve.segments[k].parent_curve)))  THEN
         RETURN (FALSE);
       END_IF;
     END_REPEAT;
     RETURN(TRUE);
   END_FUNCTION;

   FUNCTION base_axis(dim : INTEGER; axis1, axis2, axis3 : direction) :
                                                      LIST [2:3] OF
  direction;
     LOCAL
       vec    : direction;
       u      : LIST [2:3] OF direction;
       factor : REAL;
     END_LOCAL;
```

```
  IF (dim = 3) THEN
    u[3] := NVL(normalise(axis3), direction([0.0,0.0,1.0]));
    u[1] := first_proj_axis(u[3],axis1);
    u[2] := second_proj_axis(u[3],u[1],axis2);
  ELSE
    u[3] := ?;
    IF EXISTS(axis1) THEN
      u[1] := normalise(axis1);
      u[2] := orthogonal_complement(u[1]);
      IF EXISTS(axis2) THEN
        factor := dot_product(axis2,u[2]);
        IF (factor < 0.0) THEN
          u[2].direction_ratios[1] := -u[2].direction_ratios[1];
          u[2].direction_ratios[2] := -u[2].direction_ratios[2];
        END_IF;
      END_IF;
    ELSE
      IF EXISTS(axis2) THEN
        u[2] := normalise(axis2);
        u[1] := orthogonal_complement(u[2]);
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := - u[1].direction_ratios[2];
      ELSE
        u[1].direction_ratios[1] := 1.0;
        u[1].direction_ratios[2] := 0.0;
        u[2].direction_ratios[1] := 0.0;
        u[2].direction_ratios[2] := 1.0;
      END_IF;
    END_IF;
  END_IF;
  RETURN(u);
END_FUNCTION;

FUNCTION normalise (arg : vector_or_direction) : vector_or_direction;
  LOCAL
    ndim   : INTEGER;
    v      : direction;
    result : vector_or_direction;
    vec    : vector;
    mag    : REAL;
  END_LOCAL;

  IF NOT EXISTS (arg) THEN
    result := ?;

  ELSE
    ndim := arg.dim;
    IF 'SHIP_MOULDED_FORM_AIM.VECTOR' IN TYPEOF(arg) THEN
      BEGIN
        vec := arg;
        v := arg.orientation;

        IF arg.magnitude = 0.0 THEN
          RETURN(?);
        ELSE
          vec.magnitude := 1.0;
        END_IF;
      END;
    ELSE
      v := arg;
```

```
      END_IF;
      mag := 0.0;
      REPEAT  i := 1 TO ndim;
        mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
      END_REPEAT;
      IF mag > 0.0 THEN
        mag := SQRT(mag);
        REPEAT  i := 1 TO ndim;
          v.direction_ratios[i] := v.direction_ratios[i]/mag;
        END_REPEAT;
        IF 'SHIP_MOULDED_FORM_AIM.VECTOR' IN TYPEOF(arg) THEN

          vec.orientation := v;
          result := vec;
        ELSE
          result := v;
        END_IF;
      ELSE
        RETURN(?);
      END_IF;
    END_IF;
    RETURN (result);
  END_FUNCTION;

  FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
    LOCAL
      x_axis : direction;
      v      : direction;
      z      : direction;
      x_vec  : vector;
    END_LOCAL;

    IF (NOT EXISTS(z_axis)) OR (NOT EXISTS(arg)) OR (arg.dim <> 3) THEN
      x_axis := ?;
    ELSE
      z_axis := normalise(z_axis);
      IF NOT EXISTS(arg) THEN
        IF (z_axis <> direction([1.0,0.0,0.0])) THEN
          v := direction([1.0,0.0,0.0]);
        ELSE
          v := direction([0.0,1.0,0.0]);
        END_IF;
      ELSE
        IF ((cross_product(arg,z).magnitude) = 0.0) THEN
          RETURN (?);
        ELSE
          v := normalise(arg);
        END_IF;
      END_IF;
      x_vec := scalar_times_vector(dot_product(v, z), z_axis);
      x_axis := vector_difference(v, x_vec).orientation;
      x_axis := normalise(x_axis);
    END_IF;
    RETURN(x_axis);
  END_FUNCTION;

  FUNCTION cross_product (arg1, arg2 : direction) : vector;
    LOCAL
      mag    : REAL;
      res    : direction;
      v1,v2  : LIST[3:3] OF REAL;
```

```
      result : vector;
    END_LOCAL;

    IF ( NOT EXISTS (arg1) OR (arg1.dim = 2)) OR
       ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
      RETURN(?);
    ELSE
      BEGIN
        v1  := normalise(arg1).direction_ratios;
        v2  := normalise(arg2).direction_ratios;
        res.direction_ratios[1] := (v1[2]*v2[3] - v1[3]*v2[2]);
        res.direction_ratios[2] := (v1[3]*v2[1] - v1[1]*v2[3]);
        res.direction_ratios[3] := (v1[1]*v2[2] - v1[2]*v2[1]);
        mag := 0.0;
        REPEAT i := 1 TO 3;
          mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
        END_REPEAT;
        IF (mag > 0.0) THEN
          result.orientation := res;
          result.magnitude := SQRT(mag);
        ELSE
          result.orientation := arg1;
          result.magnitude := 0.0;
        END_IF;
        RETURN(result);
      END;
    END_IF;
  END_FUNCTION;

  FUNCTION scalar_times_vector (scalar : REAL; vec : vector_or_direction)
                                       : vector;
    LOCAL
      v      : direction;
      mag    : REAL;
      result : vector;
    END_LOCAL;

    IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
      result := ?;

    ELSE
      IF 'SHIP_MOULDED_FORM_AIM.VECTOR' IN TYPEOF (vec) THEN

        v   := vec.orientation;
        mag := scalar * vec.magnitude;
      ELSE
        v   := vec;
        mag := scalar;
      END_IF;
      IF (mag < 0.0 ) THEN
        REPEAT i := 1 TO SIZEOF(v.direction_ratios);
          v.direction_ratios[i] := -v.direction_ratios[i];
        END_REPEAT;
        mag := -mag;
      END_IF;
      result.orientation := normalise(v);
      result.magnitude   := mag;
    END_IF;
    RETURN (result);
  END_FUNCTION;
```

```
FUNCTION dot_product(arg1, arg2 : direction) : REAL;
  LOCAL
    scalar : REAL;
    vec1, vec2: direction;
    ndim : INTEGER;
  END_LOCAL;

  IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
    scalar := ?;

  ELSE
    IF (arg1.dim <> arg2.dim) THEN
      scalar := ?;

    ELSE
      BEGIN
        vec1   := normalise(arg1);
        vec2   := normalise(arg2);
        ndim   := arg1.dim;
        scalar := 0.0;
        REPEAT  i := 1 TO ndim;
          scalar := scalar +
                      vec1.direction_ratios[i]*vec2.direction_ratios[i];
        END_REPEAT;
      END;
    END_IF;
  END_IF;
  RETURN (scalar);
END_FUNCTION;

FUNCTION vector_difference(arg1, arg2 : vector_or_direction) : vector;
  LOCAL
    result          : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim            : INTEGER;
  END_LOCAL;

  IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <>
arg2.dim)
      THEN
    result := ?;

  ELSE
    BEGIN
      IF 'SHIP_MOULDED_FORM_AIM.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1.0;
        vec1 := arg1;
      END_IF;
      IF 'SHIP_MOULDED_FORM_AIM.VECTOR' IN TYPEOF(arg2) THEN

        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1.0;
        vec2 := arg2;
      END_IF;
      vec1 := normalise (vec1);
```

```
        vec2 := normalise (vec2);
        ndim := SIZEOF(vec1.direction_ratios);
        mag := 0.0;
        REPEAT i := 1 TO ndim;
          res.direction_ratios[i] := mag1*vec1.direction_ratios[i] -
                                     mag2*vec2.direction_ratios[i];
          mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
        END_REPEAT;
        IF (mag > 0.0 ) THEN
          result.magnitude := SQRT(mag);
          result.orientation := res;
        ELSE
          result.magnitude := 0.0;
          result.orientation := vec1;
        END_IF;
      END;
    END_IF;
    RETURN (result);
  END_FUNCTION;

  FUNCTION second_proj_axis(z_axis, x_axis, arg: direction) : direction;
    LOCAL
      y_axis : vector;
      v      : direction;
      temp   : vector;
    END_LOCAL;

    IF NOT EXISTS(arg) THEN
      v := direction([0.0,1.0,0.0]);
    ELSE
      v := arg;
    END_IF;

    temp   := scalar_times_vector(dot_product(v, z_axis), z_axis);
    y_axis := vector_difference(v, temp);
    temp   := scalar_times_vector(dot_product(v, x_axis), x_axis);
    y_axis := vector_difference(y_axis, temp);
    y_axis := normalise(y_axis);
    RETURN(y_axis.orientation);
  END_FUNCTION;

  FUNCTION orthogonal_complement(vec : direction) : direction;
    LOCAL
      result : direction;
    END_LOCAL;

    IF (vec.dim <> 2) OR NOT EXISTS (vec) THEN
      RETURN(?);
    ELSE
      result.direction_ratios[1] := -vec.direction_ratios[2];
      result.direction_ratios[2] := vec.direction_ratios[1];
      RETURN(result);
    END_IF;
  END_FUNCTION;

  FUNCTION acyclic_surface_replica(rep : surface_replica; parent :
surface)
                                                : BOOLEAN;
    IF NOT (('SHIP_MOULDED_FORM_AIM.SURFACE_REPLICA') IN TYPEOF(parent))
THEN
```

```
        RETURN (TRUE);
      END_IF;

      IF (parent :=: rep) THEN
         RETURN (FALSE);

       ELSE RETURN(acyclic_surface_replica(rep,
                       parent\surface_replica.parent_surface));
       END_IF;
     END_FUNCTION;

    FUNCTION make_array_of_array(lis : LIST[1:?] OF LIST [1:?] OF GENERIC :
   T;
                                  low1, u1, low2, u2 : INTEGER):
                     ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
      LOCAL
        n1,n2 : INTEGER;
        res   : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
        resl  : LIST[1:?] OF ARRAY [low2:u2] OF GENERIC : T;
      END_LOCAL;


      n1 := SIZEOF(lis);
      n2 := SIZEOF(lis[1]);

      IF (n1 <> (u1 -low1 + 1)) AND (n2 <> (u2 - low2 + 1)) THEN
        RETURN(?);
      END_IF;

      REPEAT i := 1 TO n1;
        IF (SIZEOF(lis[i]) <> n2) THEN
          RETURN(?);
        END_IF;
      END_REPEAT;


      REPEAT i := 1 TO  n1;
        RESL[i] := list_to_array(lis[i],low2,u2);
      END_REPEAT;

      res := list_to_array(resl,low1,u1);
      RETURN(res);
    END_FUNCTION;

    FUNCTION boolean_choose (b : boolean;
              choice1, choice2 : generic : item) : generic : item;

      IF b THEN
        RETURN (choice1);
      ELSE
        RETURN (choice2);
      END_IF;
    END_FUNCTION;

    FUNCTION path_head_to_tail(a_path : path) : LOGICAL;
      LOCAL
        n : INTEGER;
        p : BOOLEAN := TRUE;
      END_LOCAL;

        n := SIZEOF (a_path.edge_list);
```

```
     REPEAT i := 2 TO n;
       p := p AND (a_path.edge_list[i-1].edge_end :=:
                   a_path.edge_list[i].edge_start);
     END_REPEAT;

     RETURN (p);
 END_FUNCTION;

FUNCTION valid_wireframe_edge_curve (crv   : curve) : BOOLEAN;


  IF SIZEOF (['SHIP_MOULDED_FORM_AIM.LINE',
              'SHIP_MOULDED_FORM_AIM.CIRCLE',
              'SHIP_MOULDED_FORM_AIM.ELLIPSE',
              'SHIP_MOULDED_FORM_AIM.PARABOLA',
              'SHIP_MOULDED_FORM_AIM.HYPERBOLA',
              'SHIP_MOULDED_FORM_AIM.B_SPLINE_CURVE',
               'SHIP_MOULDED_FORM_AIM.POLYLINE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE

    IF ('SHIP_MOULDED_FORM_AIM.CURVE_REPLICA') IN TYPEOF (crv)
      THEN RETURN (valid_wireframe_edge_curve
                          (crv\curve_replica.parent_curve));
    ELSE

      IF ( 'SHIP_MOULDED_FORM_AIM.OFFSET_CURVE_3D') IN TYPEOF (crv)

        THEN RETURN (valid_wireframe_edge_curve
                            (crv\offset_curve_3d.basis_curve));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;

FUNCTION valid_wireframe_vertex_point (pnt   : point): BOOLEAN;

  IF ( 'SHIP_MOULDED_FORM_AIM.CARTESIAN_POINT' IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE


    IF ('SHIP_MOULDED_FORM_AIM.POINT_REPLICA') IN TYPEOF (pnt)

      THEN RETURN (valid_wireframe_vertex_point
                          (pnt\point_replica.parent_pt));
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;

 FUNCTION acyclic_point_replica(rep : point_replica; parent : point)
                                              : BOOLEAN;
   IF NOT (('SHIP_MOULDED_FORM_AIM.POINT_REPLICA') IN TYPEOF(parent))
THEN

      RETURN (TRUE);
   END_IF;

   IF (parent :=: rep) THEN
```

```
        RETURN (FALSE);

      ELSE RETURN(acyclic_point_replica(rep,
  parent\point_replica.parent_pt));
      END_IF;
    END_FUNCTION;

   FUNCTION build_axes(axis, ref_direction : direction) :
                                            LIST [3:3] OF direction;
     LOCAL
       u : LIST[3:3] OF direction;
     END_LOCAL;

     u[3] := NVL(normalise(axis), direction([0.0,0.0,1.0]));
     u[1] := first_proj_axis(u[3],ref_direction);
     u[2] := normalise(cross_product(u[3],u[1])).orientation;
     RETURN(u);
   END_FUNCTION;

  FUNCTION dimension_of(item : geometric_representation_item) :
    dimension_count;
    LOCAL
      x : SET OF representation;
      y : representation_context;
    END_LOCAL;

    -- Find the set of representation in which the item is used.

    x := using_representations(item);

    -- Determines the dimension_count of the
    -- geometric_representation_context. Note that the
    -- RULE compatible_dimension ensures that the context_of_items
    -- is of type geometric_representation_context and has
    -- the same dimension_count for all values of x.

      y := x[1].context_of_items;
      RETURN
  (y\geometric_representation_context.coordinate_space_dimension);

  END_FUNCTION;

   FUNCTION constraints_geometry_shell_based_wireframe_model
                        (m : shell_based_wireframe_model) : BOOLEAN;
     LOCAL
       result : BOOLEAN := TRUE;
     END_LOCAL;

     REPEAT j := 1 TO SIZEOF(m.sbwm_boundary);
       IF (NOT ('SHIP_MOULDED_FORM_AIM.WIRE_SHELL' IN
  TYPEOF(m.sbwm_boundary[j])) AND
          (NOT ('SHIP_MOULDED_FORM_AIM.VERTEX_SHELL' IN
  TYPEOF(m.sbwm_boundary[j]))))

       THEN
         result := FALSE;
         RETURN(result);

       END_IF;
     END_REPEAT;
```

```
   RETURN(result);
 END_FUNCTION;

FUNCTION dimensions_for_si_unit (n : si_unit_name) :
dimensional_exponents;
    CASE n OF
      metre          : RETURN (dimensional_exponents
                              (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
      gram           : RETURN (dimensional_exponents
                           (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
      second         : RETURN (dimensional_exponents
                           (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
      ampere         : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
      kelvin         : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
      mole           : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
      candela        : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
      radian         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
      steradian      : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
      hertz          : RETURN (dimensional_exponents
                           (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
      newton         : RETURN (dimensional_exponents
                           (1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
      pascal         : RETURN (dimensional_exponents
                           (-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
      joule          : RETURN (dimensional_exponents
                           (2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
      watt           : RETURN (dimensional_exponents
                           (2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
      coulomb        : RETURN (dimensional_exponents
                           (0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));
      volt           : RETURN (dimensional_exponents
                           (2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
      farad          : RETURN (dimensional_exponents
                           (-2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0));
      ohm            : RETURN (dimensional_exponents
                           (2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
      siemens        : RETURN (dimensional_exponents
                           (-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
      weber          : RETURN (dimensional_exponents
                           (2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
      tesla          : RETURN (dimensional_exponents
                           (0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
      henry          : RETURN (dimensional_exponents
                           (2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
      degree_Celsius : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
      lumen          : RETURN (dimensional_exponents
                           (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
      lux            : RETURN (dimensional_exponents
                           (-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
      becquerel      : RETURN (dimensional_exponents
                           (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
      gray           : RETURN (dimensional_exponents
                           (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
      sievert        : RETURN (dimensional_exponents
```

```
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    END_CASE;
  END_FUNCTION; -- dimensions_for_si_unit

  FUNCTION valid_units ( m : measure_with_unit ) : BOOLEAN ;
    IF 'SHIP_MOULDED_FORM_AIM.LENGTH_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.MASS_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.TIME_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.ELECTRIC_CURRENT_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.THERMODYNAMIC_TEMPERATURE_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.AMOUNT_OF_SUBSTANCE_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.LUMINOUS_INTENSITY_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.PLANE_ANGLE_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
```

```
    IF 'SHIP_MOULDED_FORM_AIM.SOLID_ANGLE_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.AREA_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.VOLUME_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.RATIO_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.POSITIVE_LENGTH_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'SHIP_MOULDED_FORM_AIM.POSITIVE_PLANE_ANGLE_MEASURE'

      IN TYPEOF ( m.value_component ) THEN
      IF derive_dimensional_exponents ( m.unit_component ) <>
         dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    RETURN (TRUE);
  END_FUNCTION; -- valid_units

  FUNCTION derive_dimensional_exponents (x : unit) :
dimensional_exponents;
    LOCAL
      result : dimensional_exponents :=
               dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    END_LOCAL;
    IF 'SHIP_MOULDED_FORM_AIM.DERIVED_UNIT' IN TYPEOF(x) THEN -- x is a
derived unit

      REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements);
        result.length_exponent                  :=
          result.length_exponent +
          (x.elements[i].exponent *
           x.elements[i].unit.dimensions.length_exponent);
```

```
          result.mass_exponent                     :=
            result.mass_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.mass_exponent);

          result.time_exponent :=
            result.time_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.time_exponent);

          result.electric_current_exponent         :=
            result.electric_current_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.electric_current_exponent);

          result.thermodynamic_temperature_exponent :=
            result.thermodynamic_temperature_exponent +
            (x.elements[i].exponent *

  x.elements[i].unit.dimensions.thermodynamic_temperature_exponent);

          result.amount_of_substance_exponent       :=
            result.amount_of_substance_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.amount_of_substance_exponent);

          result.luminous_intensity_exponent        :=
            result.luminous_intensity_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.luminous_intensity_exponent);
        END_REPEAT;
      ELSE -- x is a unitless or a named unit
        result := x.dimensions;
      END_IF;
      RETURN (result);
    END_FUNCTION; -- derive_dimensional_exponents

  RULE
  prod_def_w_class_id_class_and_statutory_designation_ref_by_exactly_one_pr
  op_def_rep_via_prop_def FOR (product_definition, property_definition,
  property_definition_representation, applied_classification_assignment);

  LOCAL
  c_a_set: SET OF applied_classification_assignment := [];
  t1_set:  SET OF product_definition := [];
  t2_set:  SET OF property_definition_representation := [];
  t3_set:  SET OF property_definition := [];
  t4_set:  SET OF product_definition := [];
  violation: BOOLEAN := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
  i.ASSIGNED_CLASSIFICATION.NAME = 'class and statutory designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
  REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
  t1_set := t1_set + c_a_set[i].items[j];
  END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
```

```
i.NAME = 'class and statutory designation');

REPEAT i := 1 TO HIINDEX(t2_set);
t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t3_set);
t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

violation := t1_set = t4_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_class_parameters_ref_by_exactly_one_prop_def_rep_via_
prop_def FOR (product_definition, property_definition,
property_definition_representation, applied_classification_assignment);

LOCAL
c_a_set: SET OF applied_classification_assignment := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF property_definition_representation := [];
t3_set:  SET OF property_definition := [];
t4_set:  SET OF product_definition := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'class parameters');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
i.NAME = 'class parameters');

REPEAT i := 1 TO HIINDEX(t2_set);
t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t3_set);
t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

violation := t1_set = t4_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_global_axis_placement_designation_ref_by_exactly_one_
prop_def_rep_via_prop_def FOR (product_definition, property_definition,
property_definition_representation, applied_classification_assignment);
```

```
LOCAL
c_a_set: SET OF applied_classification_assignment := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF property_definition_representation := [];
t3_set:  SET OF property_definition := [];
t4_set:  SET OF product_definition := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'global axis placement');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
i.NAME = 'global axis placement');

REPEAT i := 1 TO HIINDEX(t2_set);
t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t3_set);
t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

violation := t1_set = t4_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_principal_characteristics_designation_ref_by_exactly_
one_prop_def_rep_via_prop_def FOR (product_definition,
property_definition, property_definition_representation,
applied_classification_assignment);

LOCAL
c_a_set: SET OF applied_classification_assignment := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF property_definition_representation := [];
t3_set:  SET OF property_definition := [];
t4_set:  SET OF product_definition := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'principal characteristics');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
i.NAME = 'global axis placement');
```

```
REPEAT i := 1 TO HIINDEX(t2_set);
t3_set := t3_set + t2_set[i].definition;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t3_set);
t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

violation := t1_set = t4_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation
FOR (representation);

LOCAL
reps:      BAG OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['class notations hull', 'class notations
machinery'];
violation: BOOLEAN := FALSE;
END_LOCAL;

reps := QUERY(temp_rep <* representation |
SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTA
TION')

| (temp_prop_def_rep.name = 'class notation'))) > 0 );

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
rep_item.name = arg_list[j])) < 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_ship_designation_ref_by_exactly_one_apl_id_ass_w_role
_name_eq_IMO_number_or_pennant_hull_number FOR(product_definition,
applied_classification_assignment);

LOCAL
c_a_set: SET OF applied_classification_assignment := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_classification_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```

```
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_IDENTIFICATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | (i.ROLE.NAME = 'IMO
number')OR (i.ROLE.NAME = 'pennant hull number') )) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_floating_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['moulded form displacement', 'draught at
amidships', 'length of waterline', 'breadth of waterline', 'angle of
trim', 'angle of heel'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'floating position');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_property
FOR (compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['property type'];
violation: BOOLEAN := FALSE;
END_LOCAL;
```

```
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic property');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_rep_item_in_compound_rep_item_if_class_eq_offset_point_table_model
FOR (compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['offset point table type'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'offset point table model');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_rep_item_in_compound_rep_item_if_class_eq_spacing_position_with_offse
t FOR (compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
```

```
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['offset'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'spacing position with offset');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_spacing_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['position number', 'position'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'spacing position');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_stability_property FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['angle of heel', 'righting arm', 'centre of
buoyancy'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability property');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_compound_rep_item_with_class_floating_position_in_compound_rep_item_w
ith_class_hydrostatic_properties_for_constant_floating_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t3_set:  SET OF REPRESENTATION_ITEM := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic properties for constant
floating position');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'floating position');

REPEAT i := 1 TO HIINDEX(c_a_set2);
REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
t2_set := t2_set + c_a_set2[i].items[j];
END_REPEAT;
```

```
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
t3_set := t3_set + t1_set[i].item_element[j];
END_REPEAT;
violation := (SIZEOF(t3_set * t2_set) <> 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_compound_rep_item_with_class_ship_curve_in_compound_rep_item_with_cla
ss_ship_curve_segment FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t3_set:  SET OF REPRESENTATION_ITEM := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve segment');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve');

REPEAT i := 1 TO HIINDEX(c_a_set2);
REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
t2_set := t2_set + c_a_set2[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
t3_set := t3_set + t1_set[i].item_element[j];
END_REPEAT;
violation := (SIZEOF(t3_set * t2_set) <> 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_wi
```

```
th_class_spacing_position_with_offset FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t3_set:  SET OF REPRESENTATION_ITEM := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'spacing position with offset');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'spacing position');

REPEAT i := 1 TO HIINDEX(c_a_set2);
REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
t2_set := t2_set + c_a_set2[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
t3_set := t3_set + t1_set[i].item_element[j];
END_REPEAT;
violation := (SIZEOF(t3_set * t2_set) <> 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_compound_rep_item_with_class_stability_property_in_compound_r
ep_item_with_class_stability_properties_for_one_floating_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t3_set:  SET OF REPRESENTATION_ITEM := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability properties for one floating
position');

REPEAT i := 1 TO HIINDEX(c_a_set);
```

```
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability property');

REPEAT i := 1 TO HIINDEX(c_a_set2);
REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
t2_set := t2_set + c_a_set2[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
t3_set := t3_set + t1_set[i].item_element[j];
END_REPEAT;
violation := (SIZEOF(t3_set * t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_class_eq_hydrostatic_table FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['mean shell thickness'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_class_eq_midship_tumble FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
```

```
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['tumble out at bottom', 'tumble in at top',
'tumble out at side', 'tumble in at side'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'midship tumble');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_class_eq_propeller_location FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['shaft line inclination x', 'shaft line
inclination y', 'shaft line location', 'propeller location'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'propeller location');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_class_eq_ship_moulded_form_wireframe_model
FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['wireframe'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form wireframe model');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_class_eq_stability_table FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['mean shell thickness'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability table');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_hydrostatic_definition_ref_by_one_or_more_rep_w_class
_id_hydrostatic_table FOR (property_definition,
APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF representation := [];
t3_set:  SET OF property_definition_representation := [];
t4_set:  SET OF property_definition := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic definition');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t2_set);
t3_set := t3_set + USEDIN(t2_set[i],
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION');

END_REPEAT;

REPEAT i := 1 TO HIINDEX(t3_set);
t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

violation := t1_set = t4_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_stability_definition_ref_by_one_or_more_rep_w_class_i
d_stability_table FOR (property_definition,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF representation := [];
t3_set:  SET OF property_definition_representation := [];
t4_set:  SET OF property_definition := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
```

```
i.ASSIGNED_CLASSIFICATION.NAME = 'stability definition');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability table');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t2_set);
t3_set := t3_set + USEDIN(t2_set[i],
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION');

END_REPEAT;

REPEAT i := 1 TO HIINDEX(t3_set);
t4_set := t4_set + t3_set[i].definition;
END_REPEAT;

violation := t1_set = t4_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_items_with_class_hydrostatic_properties_for_constant_floating
_position_in_representation_w_class_hydrostatic_table FOR
(representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF representation := [];
t2_set:  SET OF representation_item := [];
t3_set:  SET OF representation_item := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic properties for constant
floating position');

REPEAT i := 1 TO HIINDEX(c2_a_set);
```

```
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].items);

t3_set := t3_set + t1_set[i].items[j];
END_REPEAT;
violation := (SIZEOF(t3_set* t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_items_with_class_hydrostatic_property_in_representation_w_cla
ss_hydrostatic_table FOR (representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF representation := [];
t2_set:  SET OF representation_item := [];
t3_set:  SET OF representation_item := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic table');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic property');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].items);

t3_set := t3_set + t1_set[i].items[j];
END_REPEAT;
violation := (SIZEOF(t3_set* t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
```

```
END_RULE;

RULE
one_or_more_items_with_class_in_representation_w_class_offset_point_table
_model FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF representation := [];
t2_set:  SET OF representation_item := [];
t3_set:  SET OF representation_item := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'offset point table model');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'section of offset point table');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].items);

t3_set := t3_set + t1_set[i].items[j];
END_REPEAT;
violation := (SIZEOF(t3_set* t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_items_with_class_in_representation_w_class_offset_point_table
_model2 FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF representation := [];
t2_set:  SET OF representation_item := [];
t3_set:  SET OF representation_item := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'offset point table model');
```

```
REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship point');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].items);

t3_set := t3_set + t1_set[i].items[j];
END_REPEAT;
violation := (SIZEOF(t3_set* t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_items_with_class_in_representation_w_class_offset_table_shape
_representation FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF representation := [];
t2_set:  SET OF representation_item := [];
t3_set:  SET OF representation_item := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'offset table shape representation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'offset point table model');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
```

```
REPEAT j := 1 TO HIINDEX(t1_set[i].items);

t3_set := t3_set + t1_set[i].items[j];
END_REPEAT;
violation := (SIZEOF(t3_set* t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_items_with_class_in_representation_w_class_stability_table
FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF representation := [];
t2_set:  SET OF representation_item := [];
t3_set:  SET OF representation_item := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability table');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability properties for one floating
position');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].items);

t3_set := t3_set + t1_set[i].items[j];
END_REPEAT;
violation := (SIZEOF(t3_set* t2_set) < 1);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_or_more_rep_item_in_compound_rep_item_if_class_eq_hydrostatic_propert
ies_for_constant_floating_position FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['hydrostatic property value'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hydrostatic properties for constant
floating position');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j])) <
1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_rep_item_in_compound_rep_item_if_class_eq_stability_properties_for_on
e_floating_position FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  LIST OF representation_item := [];
arg_list:  LIST OF STRING := ['centre of gravity above keel', 'definition
of starting floating position'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'stability properties for one floating
position');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
t2_set := t1_set[i].item_element;
violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
END_REPEAT;
```

```
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
four__or_more_item_in_representation_if_class_eq_ship_moulded_form_wirefr
ame_model FOR (representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['mesh definition ship curve'];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form wireframe model');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) < 4);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
two_or_more_compound_rep_item_with_class_ship_curve_in_compound_rep_item_
with_class_knot FOR (compound_representation_item,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
t3_set:  SET OF REPRESENTATION_ITEM := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'knot');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
```

```
i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve');

REPEAT i := 1 TO HIINDEX(c_a_set2);
REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
t2_set := t2_set + c_a_set2[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
t3_set := t3_set + t1_set[i].item_element[j];
END_REPEAT;
violation := (SIZEOF(t3_set * t2_set) < 2);
t3_set:= [];
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
valid_prod_def_w_classs_moulded_form_in_prod_def_w_class_ship_moulded_for
m FOR (product_definition, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF PRODUCT_DEFINITION := [];
t2_set:  SET OF PRODUCT_DEFINITION:= [];
gr_ass:    SET OF GROUP_ASSIGNMENT := [];
groups:    SET OF GROUP := [];
t3_set:  SET OF PRODUCT_DEFINITION:= [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
i.role.name = 'equivalence');
REPEAT i := 1 TO HIINDEX(gr_ass);
REPEAT j := 1 TO HIINDEX(gr_ass[i].items);
IF (gr_ass[i].items[j] IN t1_set) THEN
groups := groups + gr_ass.assigned_group;
END_IF;
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c_a_set[i].items[j];
END_REPEAT;
```

```
END_REPEAT;

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
(SIZEOF(i.items) <> 0) AND
(i.role.name = 'item structure') AND
(i.assigned_group IN groups));

REPEAT i := 1 TO HIINDEX(gr_ass);
REPEAT j := 1 TO HIINDEX(gr_ass[i].items);
t3_set := t3_set + gr_ass[i].items[j];
END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
WR1: NOT violation;
END_RULE;

RULE
valid_prod_def_rel_w_classs_moulded_form_relationship_in_prod_def_w_class
_ship_moulded_form FOR (product_definition,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
t1_set:  SET OF PRODUCT_DEFINITION := [];
t2_set:  SET OF PRODUCT_DEFINITION_RELATIONSHIP:= [];
gr_ass:    SET OF GROUP_ASSIGNMENT := [];
groups:    SET OF GROUP := [];
t3_set:  SET OF PRODUCT_DEFINITION_RELATIONSHIP:= [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship moulded form');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
i.role.name = 'equivalence');
REPEAT i := 1 TO HIINDEX(gr_ass);
REPEAT j := 1 TO HIINDEX(gr_ass[i].items);
IF (gr_ass[i].items[j] IN t1_set) THEN
groups := groups + gr_ass.assigned_group;
END_IF;
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form relationship');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t2_set := t2_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;
```

```
gr_ass := QUERY(i <* GROUP_ASSIGNMENT |
(SIZEOF(i.items) <> 0) AND
(i.role.name = 'item structure') AND
(i.assigned_group IN groups));

REPEAT i := 1 TO HIINDEX(gr_ass);
REPEAT j := 1 TO HIINDEX(gr_ass[i].items);
t3_set := t3_set + gr_ass[i].items[j];
END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_rel_w_class_moulded_form_relationship_related_prod_def_shall_hav
e_class_moulded_form FOR (product_definition_relationship,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set: SET OF PRODUCT_DEFINITION_RELATIONSHIP := [];
t2_set:  SET OF PRODUCT_DEFINITION:= [];
t3_set:  SET OF PRODUCT_DEFINITION:= [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form relationship');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := t2_set + t1_set[i].related_product_definition;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t3_set := t3_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_rel_w_class_moulded_form_relationship_relating_prod_def_shall_ha
```

```
ve_class_moulded_form FOR (product_definition_relationship,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set: SET OF PRODUCT_DEFINITION_RELATIONSHIP := [];
t2_set:  SET OF PRODUCT_DEFINITION:= [];
t3_set:  SET OF PRODUCT_DEFINITION:= [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form relationship');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := t2_set + t1_set[i].relating_product_definition;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form');

REPEAT i := 1 TO HIINDEX(c2_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t3_set := t3_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;
violation := NOT(t2_set <= t3_set);

WHERE
WR1: NOT violation;
END_RULE;

RULE
dsc_rep_item_w_n_user_def_function_if_dsc_rep_item_w_n_function_and_d_use
r_defined FOR (REPRESENTATION);

LOCAL
rep_set:   SET OF representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
(r.NAME = 'function') AND (r.DESCRIPTION = 'user defined'))) > 0) AND
(SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
(r.NAME = 'user def function'))) = 0);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;
```

```
RULE
ID_of_extended_property_definition_with_class_versionable_object_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'versionable object') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_product_with_class_definable_object_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'definable object') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE
ID_of_product_definition_with_class_class_and_statutory_designation_patte
rn FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'class and statutory designation') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_product_definition_with_class_definable_object_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
```

```
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'definable object') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_product_definition_with_class_global_axis_placement_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'global axis placement') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_product_definition_with_class_owner_designation_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'owner designation') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE
ID_of_product_definition_with_class_principal_characteristics_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'principal characteristics') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
```

```
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_product_definition_with_class_ship_designation_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'ship designation') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_product_definition_with_class_shipyard_designation_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'shipyard designation') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE
ID_of_product_definition_relationship_with_class_definable_object_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'definable object') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;
```

```
RULE ID_of_product_related_product_category_with_class_ship_type_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'ship type') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE
ID_of_property_definition_with_class_deck_moulded_form_design_parameter_p
attern FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'deck moulded form design parameter') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_property_definition_with_class_hydrostatic_definition_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'hydrostatic definition') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE
ID_of_property_definition_with_class_local_co_ordinate_system_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
```

```
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'local co-ordinate system') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE
ID_of_property_definition_with_class_moulded_form_characteristics_definit
ion_pattern FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'moulded form characteristics definition') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_property_definition_with_class_ship_overall_dimensions_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'ship overall dimensions') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_property_definition_with_class_spacing_table_pattern FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'spacing table') THEN
```

```
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE ID_of_property_definition_with_class_stability_definition_pattern
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
clfied_inst : SET OF GENERIC := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(APPLIED_CLASSIFICATION_ASSIGNMENT);
IF (APPLIED_CLASSIFICATION_ASSIGNMENT [i].assigned_classification.name =
'stability definition') THEN
clfied_inst := clfied_inst + APPLIED_CLASSIFICATION_ASSIGNMENT [i].items;
END_IF;
END_REPEAT;

WHERE
WR1: SIZEOF(QUERY(i <* clfied_inst | NOT(i.id LIKE '*\:*'))) = 0;
END_RULE;

RULE rep_has_guac FOR (REPRESENTATION);

LOCAL
v_r_i: STRING := 'SHIP_MOULDED_FORM_AIM.VALUE_REPRESENTATION_ITEM';
g_r_i: STRING := 'SHIP_MOULDED_FORM_AIM.GEOMETRIC_REPRESENTATION_ITEM';
g_u_a_c:  STRING := 'SHIP_MOULDED_FORM_AIM.GLOBAL_UNIT_ASSIGNED_CONTEXT';

has_guac: BOOLEAN := TRUE;
END_LOCAL;

REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE has_guac;
REPEAT j := 1 TO SIZEOF(REPRESENTATION[i].ITEMS) WHILE has_guac;
IF ((v_r_i IN TYPEOF(REPRESENTATION[i].ITEMS[j])) OR
(g_r_i IN TYPEOF(REPRESENTATION[i].ITEMS[j]))) THEN
has_guac := (g_u_a_c IN TYPEOF(REPRESENTATION[i].CONTEXT_OF_ITEMS));
END_IF;
END_REPEAT;
END_REPEAT;

WHERE
WR1: has_guac;
END_RULE;

RULE max_one_version_id_per_versionable_object FOR
(APPLIED_IDENTIFICATION_ASSIGNMENT);

LOCAL
version_ids: SET OF APPLIED_IDENTIFICATION_ASSIGNMENT := [];
versionable_objects: BAG OF GENERIC := [];
duplicate: BOOLEAN := FALSE;
END_LOCAL;

version_ids := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
i.ROLE.NAME = 'version identifier');
```

```
REPEAT i := 1 TO HIINDEX(version_ids);
versionable_objects := versionable_objects + version_ids[i].items;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
REPEAT j := i + 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
duplicate := versionable_objects[i] :=: versionable_objects[j];
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT duplicate;
END_RULE;

RULE id_is_globally_unique FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
duplicate: BOOLEAN := FALSE;
def_obj_cl_ass: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
clfied_inst: SET OF GENERIC := [];
END_LOCAL;

def_obj_cl_ass := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.assigned_classification.name = 'definable object');

REPEAT i := 1 TO HIINDEX(def_obj_cl_ass);
clfied_inst := clfied_inst + def_obj_cl_ass[i].items;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(clfied_inst) WHILE NOT duplicate;
REPEAT j := i + 1 TO HIINDEX(clfied_inst) WHILE NOT duplicate;
IF (clfied_inst[i].id = clfied_inst[j].id) THEN
duplicate := TRUE;
END_IF;
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT(duplicate);
END_RULE;

RULE
prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w
_class_id_class_notation FOR (PRODUCT_DEFINITION, PROPERTY_DEFINITION,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF PRODUCT_DEFINITION := [];
t2_set:  SET OF PROPERTY_DEFINITION := [];
t3_set:  SET OF PROPERTY_DEFINITION := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'class and statutory designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
```

```
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'class notation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t3_set := t3_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION' + '.' + 'DEFINITION');

END_REPEAT;
violation := t3_set <= t2_set;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_class_and_statutory_designation_ref_by_one_prop_def_w
_class_id_regulation FOR (PRODUCT_DEFINITION, PROPERTY_DEFINITION,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
c2_a_set: SET OF CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF PRODUCT_DEFINITION := [];
t2_set:  SET OF PROPERTY_DEFINITION := [];
t3_set:  SET OF PROPERTY_DEFINITION := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'class and statutory designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'regulation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
t3_set := t3_set + c2_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION' + '.' + 'DEFINITION');

END_REPEAT;
violation := t3_set <= t2_set;
```

```
WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_appendage_moulded_form_design_param
eter FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['appendage length','appendage
breadth','appendage depth','type of appendage'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'appendage moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_bottom_moulded_form_design_paramete
r FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['bilge radius','rise of floor','aft end of
flat of bottom','front end of flat of bottom','flat of bottom
breadth','length of flat of bottom breadth','rake of keel'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'bottom moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_bulb_moulded_form_design_parameter
FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
```

```
arg_list:  LIST OF STRING := ['bulb length','bulb length from pp','bulb
breadth','bulb breadth pp','bulb depth','bulb depth pp','bulb frame
section area at pp','bulb location'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'bulb moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_deck_moulded_form_design_parameter
FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['camber','sheer at AP','sheer at FP'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'deck moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_hull_moulded_form_design_parameter
FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['aft end of parallel midbody at design
draught', 'front end of parallel midbody at design draught', 'length of
parallel midbody at design draught', 'aft end of flat of side', 'front
end of flat of side', 'length of flat of side','block coefficient',
'prismatic coefficient', 'max wetted frame section area'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'hull moulded form design parameter');
```

```
REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_propeller_moulded_form_design_param
eter FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['type of propulsion', 'propeller diameter',
'chord length at 0 7 radius', 'thickness at 0 7 radius', 'number of
propeller blades', 'expanded area ratio', 'hub diameter ratio', 'nominal
design pitch ratio', 'type of propeller blades', 'rake', 'skew', 'design
sense of rotation'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'propeller moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_rudder_moulded_form_design_paramete
r FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['rudder height', 'rudder mean height',
'rudder length', 'rudder mean length', 'rudder thickness', 'projected
rudder area', 'type of the rudder', 'rudder location', 'aspect ratio'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'rudder moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;
```

```
WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_thruster_moulded_form_design_parame
ter FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['thruster tunnel diameter', 'thruster
tunnel min length', 'thruster tunnel max length', 'geometric thruster
location', 'thruster location'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'thruster moulded form design parameter');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_class_and_statutory_designation FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['class number'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'class and statutory designation');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_name_eq_class_notation FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
```

```
arg_list:  LIST OF STRING := ['service area', 'approval required for oil
cargo', 'approval required for loading unloading aground', 'approval
required for unloading grabs'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'class notation');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_name_eq_class_parameters FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['length class', 'length solas', 'scantlings
draught', 'block coefficient class', 'design speed ahead','design speed
astern','froude number'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'class parameters');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_name_eq_global_axis_placement FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['global axes and origin', 'after
perpendicular offset', 'orientation'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'global axis placement');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
```

```
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_local_co_ordinate_system_with_posit
ion_reference FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['local axes and origin'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'local co-ordinate system with position reference');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_name_eq_local_co_ordinate_system FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['local axes and origin'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'local co-ordinate system');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_moulded_form_characteristics_defini
tion FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
```

```
arg_list:  LIST OF STRING := ['moulded form outer surface', 'moulded form
displacement'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'moulded form characteristics definition');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
one_item_in_representation_if_name_eq_moulded_form_function_parameters
FOR (representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['function'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'moulded form function parameters');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_name_eq_principal_characteristics FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['length between perpendiculars', 'moulded
breadth', 'moulded depth', 'design draught', 'design deadweight','min
daught at fp','max daught at fp', 'min daught at ap','max daught at ap'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'principal characteristics');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
```

```
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE one_item_in_representation_if_name_eq_ship_overall_dimensions FOR
(representation);

LOCAL
t1_set:  SET OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['overall breadth', 'overall depth',
'overall length', 'stem overhang', 'stern overhang'];
violation: BOOLEAN := FALSE;
END_LOCAL;

t1_set:= QUERY(temp_rep <* representation |
temp_rep.name = 'ship overall dimensions');

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
rep_item.name = arg_list[j])) <> 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_class_notation
FOR (representation);

LOCAL
reps:      BAG OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['ice class notation','service factor',
'approval required for heavy cargo'];
violation: BOOLEAN := FALSE;
END_LOCAL;

reps := QUERY(temp_rep <* representation |
SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTA
TION')
| (temp_prop_def_rep.name = 'class notation'))) > 0 );

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
rep_item.name = arg_list[j])) > 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;
```

```
RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_hull_moulded_f
orm_design_parameter FOR (representation);

LOCAL
reps:      BAG OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['waterline angle of entrance at stern',
'waterline angle of entrance at bow', 'max frame section area location',
'hull length pp', 'hull breadth', 'hull depth', 'hull design draught',
'gunwale radius'];
violation: BOOLEAN := FALSE;
END_LOCAL;

reps := QUERY(temp_rep <* representation |
SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTA
TION')
| (temp_prop_def_rep.name = 'hull moulded form design parameter'))) > 0
);

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
rep_item.name = arg_list[j])) > 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_moulded_form_f
unction_parameters FOR (representation);

LOCAL
reps:      BAG OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['user def function'];
violation: BOOLEAN := FALSE;
END_LOCAL;

reps := QUERY(temp_rep <* representation |
SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTA
TION')
| (temp_prop_def_rep.name = 'moulded form function parameters'))) > 0 );

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
rep_item.name = arg_list[j])) > 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_principal_char
acteristics FOR (representation);
```

```
LOCAL
reps:      BAG OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['block coefficient'];
violation: BOOLEAN := FALSE;
END_LOCAL;

reps := QUERY(temp_rep <* representation |
SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTA
TION')
| (temp_prop_def_rep.name = 'principal characteristics'))) > 0 );

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
rep_item.name = arg_list[j])) > 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
zero_or_one_items_in_rep_if_used_in_prop_def_rep_w_name_eq_propeller_moul
ded_form_design_parameter FOR (representation);

LOCAL
reps:      BAG OF REPRESENTATION := [];
arg_list:  LIST OF STRING := ['blade mean height'];
violation: BOOLEAN := FALSE;
END_LOCAL;

reps := QUERY(temp_rep <* representation |
SIZEOF (QUERY(temp_prop_def_rep <* USEDIN(temp_rep,
'SHIP_MOULDED_FORM_AIM.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTA
TION')
| (temp_prop_def_rep.name = 'propeller moulded form design parameter')))
> 0 );

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
rep_item.name = arg_list[j])) > 1);
END_REPEAT;
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
comp_rep_item_w_class_id_section_of_offset_point_table_ref_by__one_class_
ass_w_role_section_identifier FOR(compound_representation_item,
applied_classification_assignment, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF compound_representation_item := [];
t2_set:  SET OF applied_classification_assignment := [];
```

```
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'section of offset point table');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_CLASSIFICATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'section
identifier')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_managin
g_company FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_organization_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'owner designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'managing
company')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_orderin
g_company FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_organization_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'owner designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'ordering
company')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_owner_designation_ref_by_one_organ_ass_w_role_owning_
company FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_organization_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'owner designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'owning
company')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;
```

```
RULE
prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_call_sig
n FOR(product_definition, applied_classification_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_classification_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_CLASSIFICATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'call sign'))
= 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_flag_sta
te FOR(product_definition, applied_classification_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_classification_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_CLASSIFICATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'flag state'))
= 1);
```

```
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_ship_designation_ref_by_one_class_ass_w_role_port_of_
registration FOR(product_definition, applied_classification_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_classification_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_CLASSIFICATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'port of
registration')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prod_def_w_class_id_shipyard_designation_ref_by_one_organ_ass_w_role_ship
yard FOR(product_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF product_definition := [];
t2_set:  SET OF applied_organization_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'shipyard designation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
```

```
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'shipyard')) =
1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_appendage_moulded_form_design_parameter_ref_by_one_pr
op_def_rep_w_name_appendage_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'appendage moulded form design
parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'appendage moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_bottom_moulded_form_design_parameter_ref_by_one_prop_
def_rep_w_name_bottom_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'bottom moulded form design parameter');
```

```
REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'bottom moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_bulb_moulded_form_design_parameter_ref_by_one_prop_de
f_rep_w_name_bulb_moulded_form_design_parameter FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'bulb moulded form design parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'bulb moulded form
design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_deck_moulded_form_design_parameter_ref_by_one_prop_de
f_rep_w_name_deck_moulded_form_design_parameter FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
```

```
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'deck moulded form design parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'deck moulded form
design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_hull_moulded_form_design_parameter_ref_by_one_prop_de
f_rep_w_name_hull_moulded_form_design_parameter FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'hull moulded form design parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'hull moulded form
design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_propeller_moulded_form_design_parameter_ref_by_one_pr
op_def_rep_w_name_propeller_moulded_form_design_parameter
```

```
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'propeller moulded form design
parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'propeller moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_rudder_moulded_form_design_parameter_ref_by_one_prop_
def_rep_w_name_rudder_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'rudder moulded form design parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'rudder moulded
form design parameter')) = 1);
END_REPEAT;
```

```
WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_pro
p_def_rep_w_name_thruster_moulded_form_design_parameter
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'thruster moulded form design
parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'thruster moulded
form design parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_class_notation_ref_by_one_organ_assw_role_class_socie
ty FOR(property_definition, applied_organization_assignment,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF applied_organization_assignment := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'class notation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;
```

```
REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'APPLIED_ORGANIZATION_ASSIGNMENT' + '.' + 'ITEMS');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.ROLE.NAME = 'class
society')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_class_notation_ref_by_one_prop_def_rep_w_name_class_n
otation FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'class notation');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'class notation'))
= 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_local_co_ordinate_system_with_position_reference_ref_
by_one_prop_def_rep_w_name_local_co_ordinate_system_with_position_referen
ce FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'local co-ordinate system with position
reference');
```

```
REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'local co-ordinate
system with position reference')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_local_co_ordinate_system_ref_by_one_prop_def_rep_w_na
me_local_co_ordinate_system FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'local co-ordinate system');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'local co-ordinate
system')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_moulded_form_characteristics_definition_ref_by_one_pr
op_def_rep_w_name_moulded_form_characteristics_definition
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
```

```
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form characteristics
definition');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'moulded form
characteristics definition')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_ship_overall_dimensions_ref_by_one_prop_def_rep_w_nam
e_ship_overall_dimensions FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'ship overall dimensions');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'ship overall
dimensions')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_local_co_ordinate_system_ref_by_one_item_defined_tran
```

```
sformation_w_name_transformation_to_parent FOR(property_definition,
item_defined_transformation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF item_defined_transformation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'local co-ordinate system');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'ITEM_DEFINED_TRANSFORMATION' + '.' + 'TRANSFORM_ITEM_1');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'transformation to
parent')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_moulded_form_functional_definition_ref_by_one_prop_de
f_rep_w_name_moulded_form_function_parameters FOR(property_definition,
property_definition_representation, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_representation := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'moulded form functional definition');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'moulded form
function parameters')) = 1);
END_REPEAT;

WHERE
```

```
WR1: NOT violation;
END_RULE;

RULE
prop_def_w_class_id_thruster_moulded_form_design_parameter_ref_by_one_pro
p_def_rel_w_name_thruster_propeller_parameter FOR(property_definition,
property_definition_relationship, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
t1_set:  SET OF property_definition := [];
t2_set:  SET OF property_definition_relationship := [];
violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'thruster moulded form design
parameter');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
t1_set := t1_set + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set);
t2_set := USEDIN(t1_set[i], 'SHIP_MOULDED_FORM_AIM.' +
'PROPERTY_DEFINITION_RELATIONSHIP' + '.' +
'RELATING_PROPERTY_DEFINITION');

violation := NOT (SIZEOF(QUERY(i <* t2_set | i.name = 'thruster propeller
parameter')) = 1);
END_REPEAT;

WHERE
WR1: NOT violation;
END_RULE;

RULE components_related_to_address FOR (EXTERNAL_SOURCE);

LOCAL
parm_list: LIST OF STRING := ['internal location', 'street number',
'street', 'postal box', 'town', 'region', 'postal code', 'country',
'facsimile number', 'telephone number', 'electronic mail address', 'telex
number'];
addr_list: SET OF EXTERNAL_SOURCE;
comp_list: SET OF EXTERNAL_SOURCE;
addr_relationships: SET OF EXTERNAL_SOURCE_RELATIONSHIP := [];
comp_relationships: SET OF EXTERNAL_SOURCE_RELATIONSHIP := [];
found: BOOLEAN := FALSE;
c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.ASSIGNED_CLASSIFICATION.NAME = 'address');

REPEAT i := 1 TO HIINDEX(c_a_set);
REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
addr_list := addr_list + c_a_set[i].items[j];
END_REPEAT;
END_REPEAT;
```

```
REPEAT i:=1 TO HIINDEX(addr_list);
addr_relationships := addr_relationships +
USEDIN(addr_list[i],'SHIP_MOULDED_FORM_AIM.EXTERNAL_SOURCE_RELATIONSHIP.R
ELATED_SOURCE');
END_REPEAT;

REPEAT i:=1 TO HIINDEX(parm_list) WHILE (NOT found);
comp_list := QUERY(temp <* EXTERNAL_SOURCE
| temp.description = parm_list[i] );
END_REPEAT;

REPEAT j:=1 TO HIINDEX(comp_list) WHILE (NOT found);
comp_relationships := USEDIN(comp_list[j],
'SHIP_MOULDED_FORM_AIM.EXTERNAL_SOURCE_RELATIONSHIP.RELATING_SOURCE');

found := SIZEOF(addr_relationships * comp_relationships) = 0 ;
END_REPEAT;

WHERE
WR1: NOT found;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_ship_point FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['point shape'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship point');

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_ship_curve FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
```

```
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['side condition', 'curve shape'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve');

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;

RULE one_rep_item_in_compound_rep_item_if_class_eq_ship_surface FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['surface shape'];
    violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship surface');

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set | items.name =
    arg_list[j])) <> 1);
    END_REPEAT;
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;

RULE
one_compound_rep_item_with_class_spacing_position_in_compound_rep_item_wi
```

```
th_class_ship_curve_with_spacing_position FOR
(compound_representation_item, APPLIED_CLASSIFICATION_ASSIGNMENT);

LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    violation: BOOLEAN := FALSE;
END_LOCAL;

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'ship curve with spacing
      position');

REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
          t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
END_REPEAT;

c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.ASSIGNED_CLASSIFICATION.NAME = 'spacing position');

REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
          t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
END_REPEAT;

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + t1_set[i].item_element[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
END_REPEAT;

WHERE
    WR1: NOT violation;
END_RULE;


END_SCHEMA; -- ship_moulded_form_AIM
```

Annex B
(normative)

AIM short names

The following table provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of short names are found in the implementation methods included in ISO 103030.

Will be provided later.

Annex C
(normative)

Implementation method specific requirements

There are no implementation method-specific requirements for this part of ISO 10303.

Annex D
(normative)

Protocol Implementation Conformance Statement (PICS) proformaAnnex D Protocol Implementation Conformance Statement (PICS) proforma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preperation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

Annex E
(normative)

Information object registrationAnnex E Information object registration

E.1 Document identification

To provide for unambiguous identification of an information object in an open system, the object identifier

{iso standard 10303 part(216) version(-1)}

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

E.2 Schema identification

To provide for unambiguous identification of the ship_moulded_form_schema in an open system, the object identifier

{iso standard 10303 part(216) version(-1) object(1) ship-moulded-form-schema(1)}

is assigned to ship_moulded_form_schema expanded schema (see annex A).

The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex F
(informative)

Application activity model

The Application activity model (AAM) is provided as an aid to understand the scope and information requirements defined in this application protocol. The model is presented as a set of figures that contain the activity diagrams and a set of definitions of the activities and their data. The application activity model is given in Figure 54 through Figure 67. Activities and data flows that are out of scope are marked with an asterisk.

NOTE - The viewpoint of the application activity model is an observer of the global ship development process. This activity model identifies the life cycle activities across all shipbuilding AP's. Activities related to the shipbuilding lifecycle that are not expanded in this activity model are detailed in other shipbuilding application protocols.

F.1 Application activity model definitions and abbreviations

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol.

The definitions given in this annex do not supersede the definitions given in the main body of the text.

approved design: The approved design is the final design to be submitted as an offer.

arrangements*: The arrangements of the ship are the ship's compartments and spaces. Any description of arrangements will include associated definitions of purpose for the compartment or space.

assemble ship*: The activity that assembles the modular units, the serviced parts and additional material that result from the production of steel sub-section. The result is an assembled ship that still has to be tested.

availability, reliability and maintainability information*: The information about the components that is required to install them in the ship and is required for planned maintenance.

basic hull parameters: Estimated principal dimensions based on historical data or preliminary design development.

budget*: The cost constraint on the design building and maintenance of the ship.

calculate CFD*: Calculation of flow around the hull. The results contribute to the hydrodynamics and powering requirements calculations.

calculate cost of ship*: This activity describes creation of negotiating documents based on technical product data and their estimated manufacturing cost. The results of this activity may contain sale price documents, financing support plan and documents describing funding and possible loans.

calculate initial hydrostatic properties: The activity that describes the initial calculation of the hydrostatic properties that, if unsatisfactory, may necessitate modifications to the hull moulded form shape.

certificates*: The certificates issued by the Classification Society on completing the ship.

check design against rules and regulations: This is the top level activity for the approval of the primary design as part of the approval and certification process. The content of this activity is the same for all ships when it comes to conformance with Main Class Rules, but varies when it comes to additional class rules (type of vessel) and register notations. The activities performed are tailored to the rule requirements for general arrangement and global strength. This part of the approval is necessary before the yard can start ordering steel.

Classification Society: An organization that enhances the safety of life and property at sea by providing rules, regulations and personnel for assessing and classifying ships during their lifecycle.

complete and approve design of machinery*: The selection, arrangement and approval of the power plant in terms of the main engine, associated propulsion system and its auxiliary machinery.

complete and approve design of outfitting and distribution systems*: The selection and approval of the necessary outfitting equipment. The selection is based mainly on former designs and in accordance with the requirements. It also contains the layout of the different types of distribution systems such as piping and HVAC.

complete and approve design of ship structure*: The completion and approval of the ship structural design.

complete and approve ship design: The production and approval of ship design product data, documents and the classification drawings using the preliminary design from the bid preparation, as well as the required rules and regulations. The result of this activity is the approved design and the production and delivery schedule.

completed sectional area curve: The curve created by plotting the areas of cross-sections through the hull at specified intervals. A completed sectional area curve is used to assess the suitability of the hull moulded form.

consultants: Organizations that provide specific services to shipyards, ship owners and classification societies during the ship lifecycle.

contract: The contract is the output from the activity which involves placing the order for the ship. The contract is used as a constraint in subsequent activities such as final design and approval and production.

cost*: The calculated cost of the ship based on the cost of material and labour.

create preliminary design: All design activities relevant in a very preliminary stage of ship design in consideration of classification rules, national/international demands, shipyard constraints and owner requirements. The aim of this task is to make a shipyard offer.

create preliminary general arrangements*: The activity that produces the preliminary compartmentation plans from the preliminary hull form definition.

create preliminary hull form: The activity that is the first step of designing a ship. Using parent ships main dimensions and form parameters one or more preliminary hull forms will be generated.

create preliminary machinery design*: The activity that produces the preliminary designs for the ship machinery; including the prime mover, shaft system, fuel system, power systems and cargo handling equipment.

create preliminary outfitting design*: The activity that produces the preliminary design for the ship's outfitting, including distributed systems, such as piping and electrical systems.

create preliminary structure design*: The activity that produces the preliminary steel structure design, including the arrangement of the primary structural members.

decide post-sales & maintenance support*: The activity that puts together the maintenance package for the ship. This is part of the tender document and includes the post sales support.

decommission and disassemble*: All activities relating to the last stage of the ship's lifecycle. It consists of the decommissioning and dismantling of the ship.

design schedule: Data that controls the time from the design phase to production.

distribution and outfitting design*: The design of the distribution systems (electrical and piping) and the outfitting.

do parametric variations: The activity that creates variations of preliminary hull moulded form solutions by changing hull moulded form parameters.

estimate form parameters: The activity that deals with a preliminary establishment of form parameters with the help of historical ship data from parent ships and estimated main dimensions.

estimate hydrodynamics and powering*: The activity that approximates hydrodynamic properties data calculations such as resistance, propulsion, seakeeping and manoeuvrability for the preliminary hull form.

estimate main dimensions and parameters: The activity that deals with the preliminary establishment of main dimensions and related parameters with the help of historical ship data.

estimate manoeuvrability*: The activity that deals with approximating the manoeuvrability of the ship and comparison of the results with design requirements. The proof of the ship's manoeuvrability will principally be given in practice or by model testing. Measuring of rudder forces and rudder moments as well as the radius of the turning circle during model tests will be done either in circulating water channels or manoeuvring basins.

estimate resistance and powering*: The activity that includes the calculations based on historical data for producing powering and resistance data for the initial preliminary design.

estimate sea-keeping*: The activity that calculates the theoretical behaviour of a ship in seaway. The solution of the coefficients of equations of motion may be obtained either by analytical or numerical methods. The natural periods of the ship will be calculated for the rolling, pitching and heaving motions.

evaluate request & schedule bid: This describes the activities of the shipyard when evaluating the inquiry of the ship owner for a new ship.

fair hull: The activity that deals with the adjustment of the hull moulded form to correct any surface aberrations.

feedback: The outputs from activities which then feed back and modify previous activities in the lifecycle on the current or subsequent ships.

finalise and approve general arrangements*: The activity that details the general arrangement after having created a draft layout. The ship's systems are described by a compartment and access drawing, showing the location, the access and the size of the different compartments.

finalise and approve hull form: The activity in which the hull form is finalised from the preliminary design. The result is a final and approved hull form design.

finalise and approve hydrodynamics and powering*: This includes all relevant hydrodynamic calculations such as resistance, propulsion, seakeeping and manoeuvrability.

finalise main dimensions and parameters: The activity that refines the completion of the main parameters and form parameters.

fore-body definition: The hull moulded form design of the fore body of the ship.

general arrangements*: The space arrangement plan from the preliminary design stage.

generate initial aft-body definition: **T**he activity that generates the hull moulded form of the aft-body of the ship from the main dimensions and form parameters taking into consideration the number and size of propellers.

generate initial deck definition: The activity that generates the position and shape of the main weather deck of the ship from the main dimensions and form parameters.

generate initial fore-body definition: The activity that generates the initial hull moulded form of the fore-body of the ship from the main dimensions and form parameters.

generate initial hull form definition: The activity that generates the initial hull moulded form shape based on the calculated main dimensions and form parameters.

generate initial mid-body definition: The activity that generates the initial hull moulded form of the mid-body of the ship from the main dimensions and form parameters.

historical data from previous designs: Data held by the shipyard or model basin on previous ship designs and used to estimate the hydrodynamics, powering requirements and sea-keeping.

hull form parameters: The block coefficient, prismatic coefficient and Froude number that are used during the preliminary design phase to estimate the hydrostatic and hydrodynamic properties of the ship.

hull form sections: The design of the hull moulded form at planar sections taken along the longitudinal axis of the ship.

hull moulded form with deck definition: The shape of the hull moulded form and enclosing watertight deck of the ship.

hull moulded form: The definition of the shape of the hull of the ship, resulting from the addition of the aft-body, mid-body and fore-body definitions, which does not take into account the thickness of the material from which the hull is made.

hydrodynamics & powering results*: The results of calculations and model basin tests. They contain resistance, propulsion, propeller performance, brake power, service speed, sea keeping and manoeuvrability data.

hydrostatics table: The data, in tabular format, that describes the hydrostatic properties of the ship which result from calculations at the initial and final design stages.

knowledge and experience: The previous experience and knowledge of companies involved throughout the ship lifecycle.

laws, rules and regulations: National laws, statutory regulations and classification society rules that are used to control the design, manufacture, operation, maintenance and scrapping of the ship.

list of required certificates*: The result of placing an order, this is the list supplied by the owner for certificate requirements.

loading and stability manual: A booklet which is placed on board the ship for the information of the master, which enables him or her to load the ship within prescribed limits, relating to strength and stability.

loading conditions*: The standard loading conditions, defining the quantities of cargo, ballast water and consumables such as fuel oil and lubrication oil, in each space or compartment, that are used as a basis for design.

machinery design*: The design drawings and electronic models of the ship mechanical systems. An output from the final design process.

machinery weights: These outputs are the result of several calculation and design activities which result in an estimated weight for all machinery.

main dimension: The length, breadth, depth and draught of the ship that are estimated during the preliminary hull moulded form design and then finalised during the main design phase.

manufacturing restrictions: A constraint of the ship construction and design processes governed by available technology and shipyard facilities.

material list*: The list of raw materials needed to manufacture the ship. A result of the final design process.

mid-body definition: The hull moulded form design of the mid body of the ship.

model basin consultants: An organization that perform model basin tests to calculate hydrodynamics and powering data.

model basin theory*: The theory, along with empirical data, used by the model basin consultants to calculate the hydrodynamics and powering information.

modifications from machinery*: Modifications to the hydrodynamics and powering due to feedback from the preliminary machinery design.

modifications to hull form: Modifications to the hull shape due to feedback from hydrodynamics and powering results and the final design process.

modifications to hull form parameters: Modifications to the form parameters due to the result of performing parametric variations in the preliminary design stage.

modifications to main dimensions: Changes to the main dimensions of the hull that are made during the hull moulded form design process.

modular units*: Sub-sections of the ship complete with machinery and outfitting which will be assembled to create the final product.

offer: The result of the preliminary design process. It will contain the shipyard's data for producing the requested ship.

offer guidelines: The offer guidelines include the data necessary to make an unconditional offer to the ship owner

operate and maintain a ship*: The activity that describes the running and maintenance of the ship during its service lifetime.

operational information*: Accumulated information during the operation phase of the ship used for maintenance and in the final scrapping stage.

owner: The organization which requests, orders, takes delivery of, and for the purposes of this model, operates the ship.

owner request, requirements: The requirements document that is submitted to the shipyard by the owner upon the invitation to tender.

operate and maintain a ship*: The activity of operating and maintaining the ship during its service lifetime.

operational information*: Accumulated information during the operation phase of the ship used for maintenance and in the final scrapping stage.

owner request, requirements: The requirements document that is submitted to the shipyard by the owner upon the invitation to tender.

owner: The organization that requests, orders and takes delivery of the ship.

perform model tests*: The activity that contributes to the hydrodynamics and powering results by the means of model tests in a towing tank or model basin.

perform ship lifecycle: All of the lifecycle activities which are associated with a ship.

place order*: The owner places an order for a ship from the bids that have been submitted. From this a contract is awarded.

planned maintenance system*: Data created during the final design process and used during the operation and maintenance of the ship.

power requirements for engine*: The hydrodynamics and powering data used in the selection of the main engine.

pre layout: The very initial layout of the ship which is produced during the bid evaluation stage and is the basis for the preliminary design.

predict brake power and service speed*: The activity that estimates the brake power from the engine and the delivered power at the propeller in order to be able to choose the correct size of the main engine.

predict propeller performance*: The activity that uses propulsion data to produce a preliminary propeller design.

predict propulsion data*: An activity that calculates the propulsion of the ship. An approximate definition of driving power can be made using the displacement equation for specific Froude numbers in the preliminary design stage.

predict resistance*: The activity that predicts the resistance of the ship. The resistance calculation uses historical data related to the hull moulded form geometry.

preliminary design: The preliminary design is one that is completed in the phases leading up to the submission of the tender.

preliminary hull form: The definition of the hull moulded form resulting from the preliminary design process that is used in the offer documents and for preliminary hydrodynamics and powering calculations.

preliminary general arrangements: The definition of the ship general arrangements as a result of the preliminary design process.

preliminary hull form: The definition of the hull form, as a result of the preliminary design process. Used in the offer documents and for preliminary compartment design, hydrodynamics and powering calculations.

preliminary machinery design*: The definition of the ship's mechanical systems. Used early to estimate the noise, speed and vibration and to estimate the machinery weights.

preliminary machinery, structure and outfitting design*: Feedback consisting of the preliminary designs for machinery, structure and outfitting and furnishing. This allows the creation of preliminary general arrangements.

preliminary outfitting design*: The definition of the ship's outfitting and accommodation, resulting from the preliminary design process.

preliminary structure design*: The definition of the preliminary ship structure during the preliminary design process.

prepare bid: This activity includes all activities of the yard regarding preparation and submission of the offer to the ship owner for the ship to be built.

present offer*: The activity concerned with presentation of the offer to build the ship to the prospective ship owner.

produce and approve reference documents: The technical documentation for the ship is produced using production information. The output includes the loading and stability manual that is approved by the Classification Society.

produce and inspect a ship: This activity includes high-level activities such as produce, monitor and inspect ship production. Inspect means, the controlling of all activities throughout the whole production life cycle of a ship.

produce modular build units*: This activity covers the production of the modular units which will make up the completed ship. They are produced from the steel-subsections and their production is controlled by the schedule, contract, the approved design, and any manufacturing restrictions. The results of the activity are the modular units, which are assembled into the ship.

produce steel sub-sections*: This activity covers the production of the steel sub-sections which make up the structure of the completed ship. This is controlled by the schedule, contract, the approved design, and any manufacturing restrictions.

product component information*: The technical data about the components that will be incorporated into the ship. These are taken into consideration when the preliminary designs are being made.

production and delivery schedule: The schedule according to which the ship is manufactured and delivered.

production information: Information describing a product e.g., dimensions, mechanical properties, workshop information.

propeller design*: The design of the propeller or propulsor as a result of the hydrodynamics and powering calculations. The design controls some of the machinery design activity.

propeller theory*: The body of knowledge based on the historical experience and abstract ideas of ship propeller performance.

quality assurance: The rules applied by an organization within the shipyard that has the task to audit the shipyard organization and applied processes in a manner such that the quality of the resulting product is assured.

regular wave theory*: The body of knowledge relating to the motion response of a ship in waves of constant height and period.

request a ship: The first activities of a ship owner when intending to order a ship. Having definite ideas regarding appearance and functionality of the ship, the owner expresses these ideas in an inquiry to the shipyard.

request for production changes: Changes that are requested to the ship design as a result of production experience or difficulties with the realisation of the ship design.

resistance and shaft power*: The result of the activity to estimate hydrodynamics and powering. Resistance and shaft power is a constraint on the creation of the preliminary hull form.

resistance theory*: The theory used to predict the resistance of the hull to forward motion in the sea.

results of zigzag, initial turning etc.*: The manoeuvring test results from model basin tests or sea trials.

resources: The shipyard, classification society, and outside consultants.

schedule: The schedule is formed as a part of the final design process. It governs the timing of the production phases.

scrapping plan*: The document used to schedule the time and resources required to dismantle the ship.

ship: A large waterborne vessel whose design, manufacture and lifecycle operation is governed by the principles of naval architecture and in accordance with international and classification society regulations.

ship product model data: The product data of the accumulated throughout its lifecycle. Since scrapping is part of the lifecycle the ship is not an output, only the documented information and knowledge about the ship survives.

ship weight modifications*: Modifications to ship weight due to the preliminary structure design. This is fed back to modify the preliminary hull form and revise the preliminary general arrangements.

shipyard: An organization that designs, builds, maintains, and repairs ships.

short and long term responses*: The results of estimation of sea keeping ability that are based on regular waves of a specified height and period and are used to predict the maximum response in irregular waves over a long period of time.

specify ship: All activities associated with the production of a detailed specification of the ship prior to a contract being placed.

steel sub-sections*: The sub-sections of the steel structure which are outfitted with the machinery and distribution systems before assembly.

structural design*: The design of the hull structure including hull, bulkheads, decks and stiffeners.

technical documentation: In case of maintenance the technical documentation of a system means part of the product description required to perform preventative maintenance, repair and failure analysis of that system. Technical information is an output, which includes more detail information about material parts needed for producing the ship/system.

technical requirements: The owner’s specifications that must be realised by the completed ship.

test results*: Maintenance test results are the results of functional tests carried out after the execution of maintenance actions.

test ship: This activity tests the actual ship against the design, contract and rules and regulations. The structure is tested and sea trials are carried out. The test results are an output from this activity.

test structures: The steel structures are tested against rules and regulations and the design. The output is the test result documentation.

test systems*: The ship’s systems including outfitting, machinery and mission systems are tested against rules and regulations and the design. The output is the test result documentation.

total resistance*: The total resistance of the hull due to forward motion in the water.

towing tank model: A model of the ship used in a towing tank for estimating hydrodynamic properties.

transportation need: A constraint which determines the specification for the ship construction.

weights and centres of gravity*: Weights and centres of gravity necessary for further calculations.

workload*: The total effort required to build the chosen ship design as estimated by the shipyard and assisting consultants.

F.2 Application activity model diagrams

The application activity model diagrams are given in Figure 54 through Figure 67. The graphical form of the Application Activity Model is presented in the IDEF0 [1] activity modelling format. Activities and data flows that are out of scope are marked with asterisks.

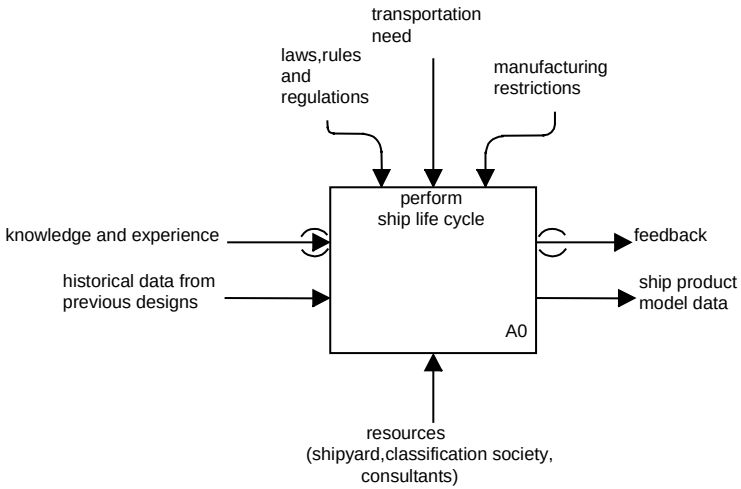


Figure 54 - Node A0 moulded form lifecycle description

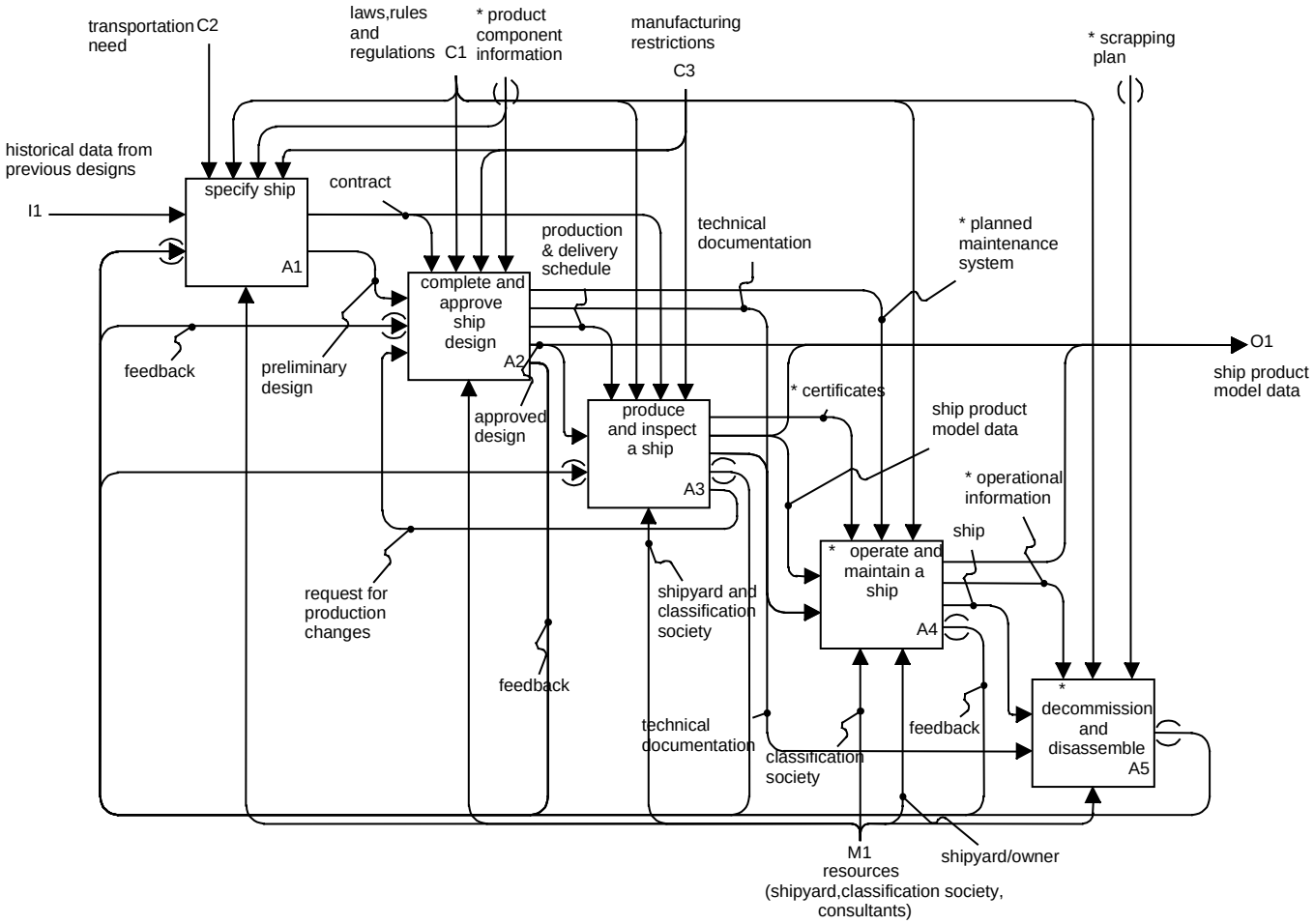


Figure 55 - Node A0 perform ship life cycle

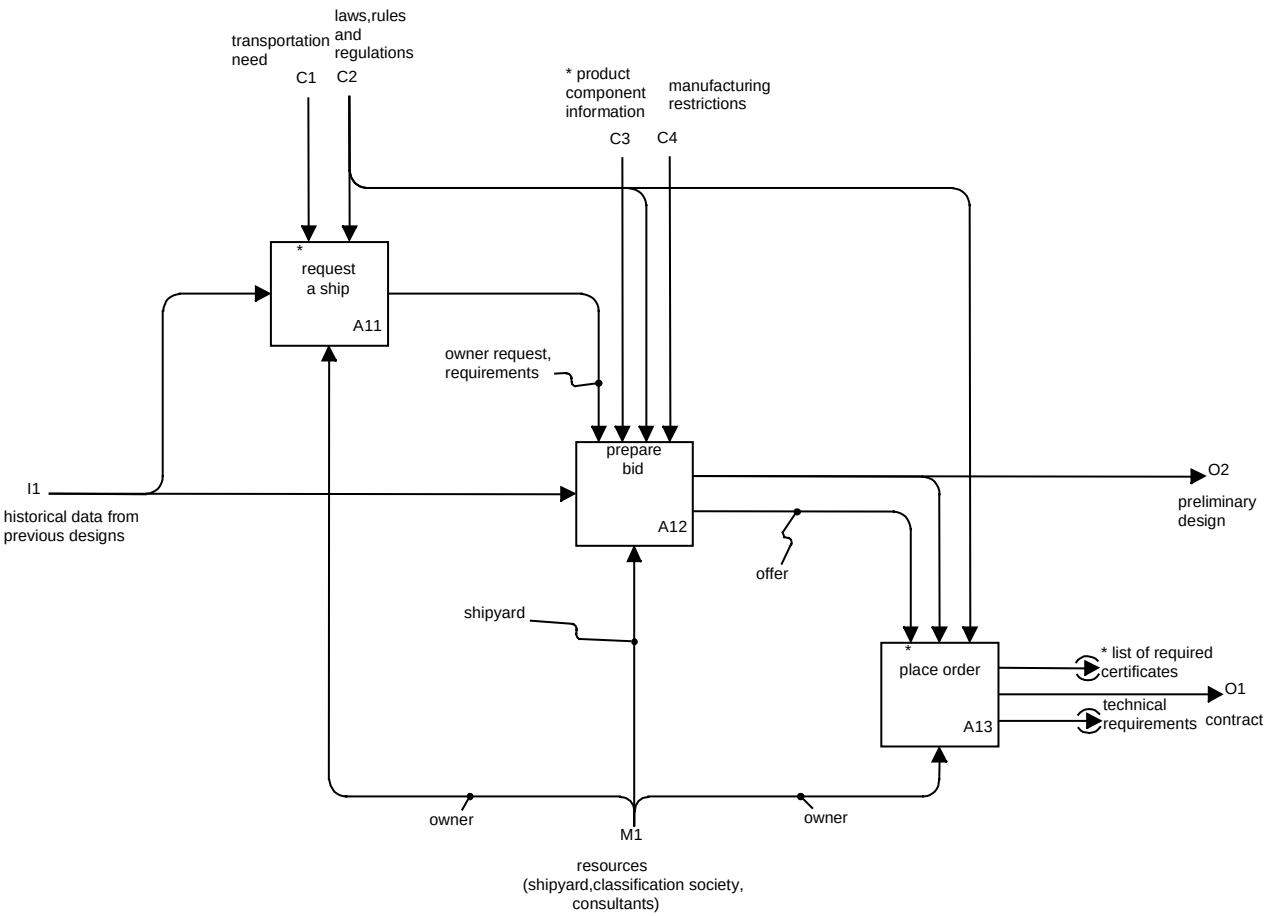


Figure 56 - Node A1 specify ship

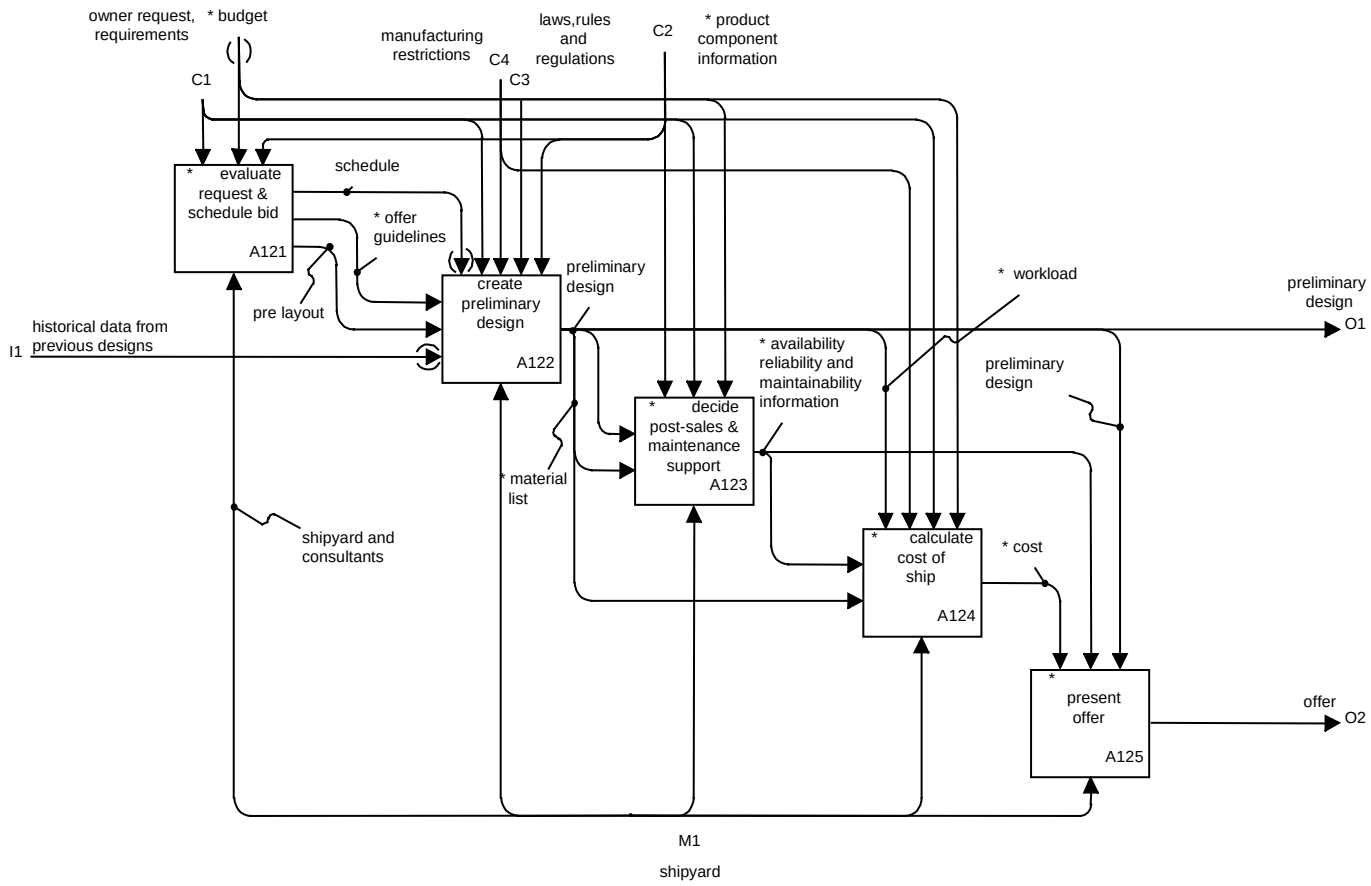


Figure 57 - Node A12 prepare bid

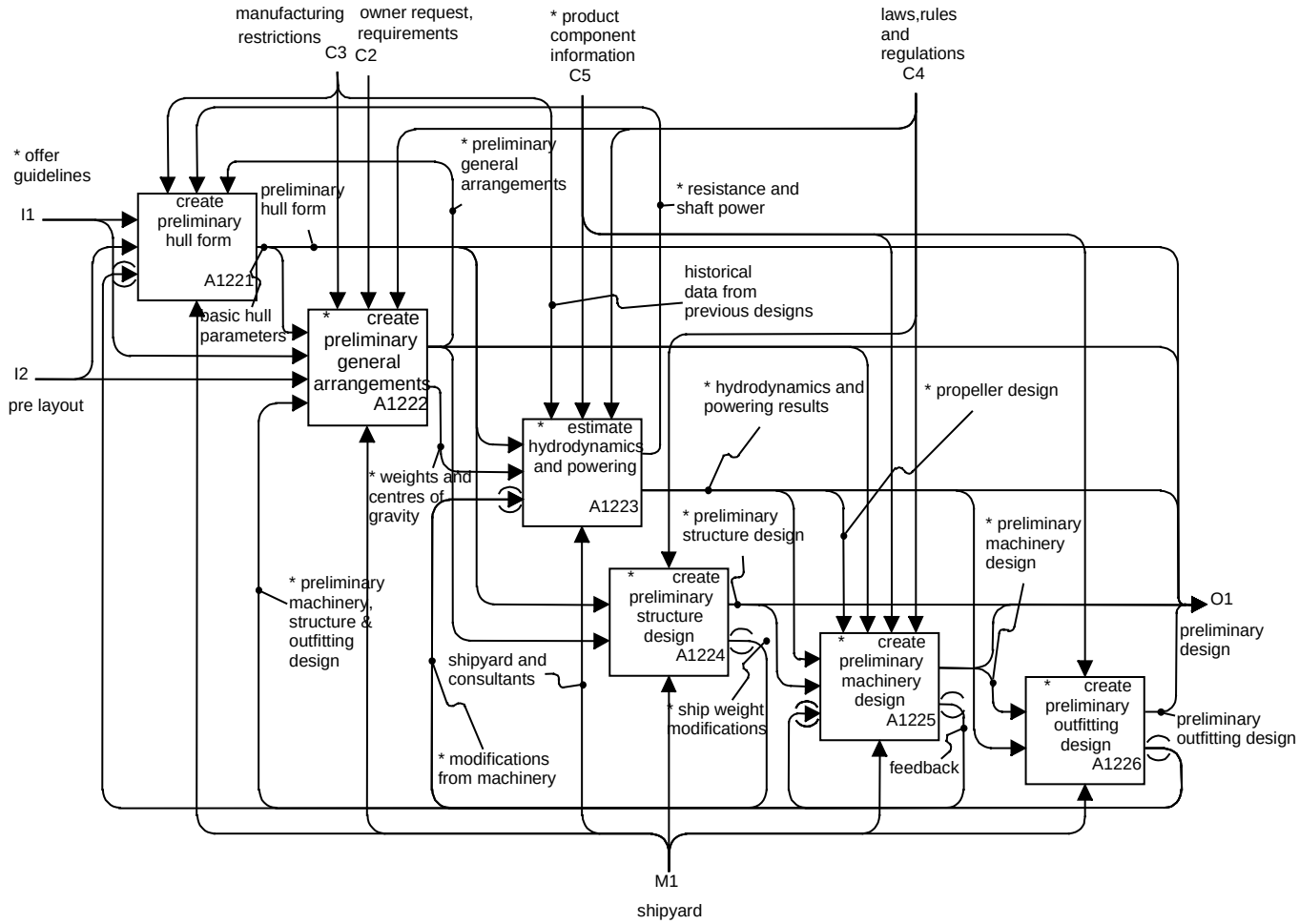


Figure 58 - Node A122 create preliminary design

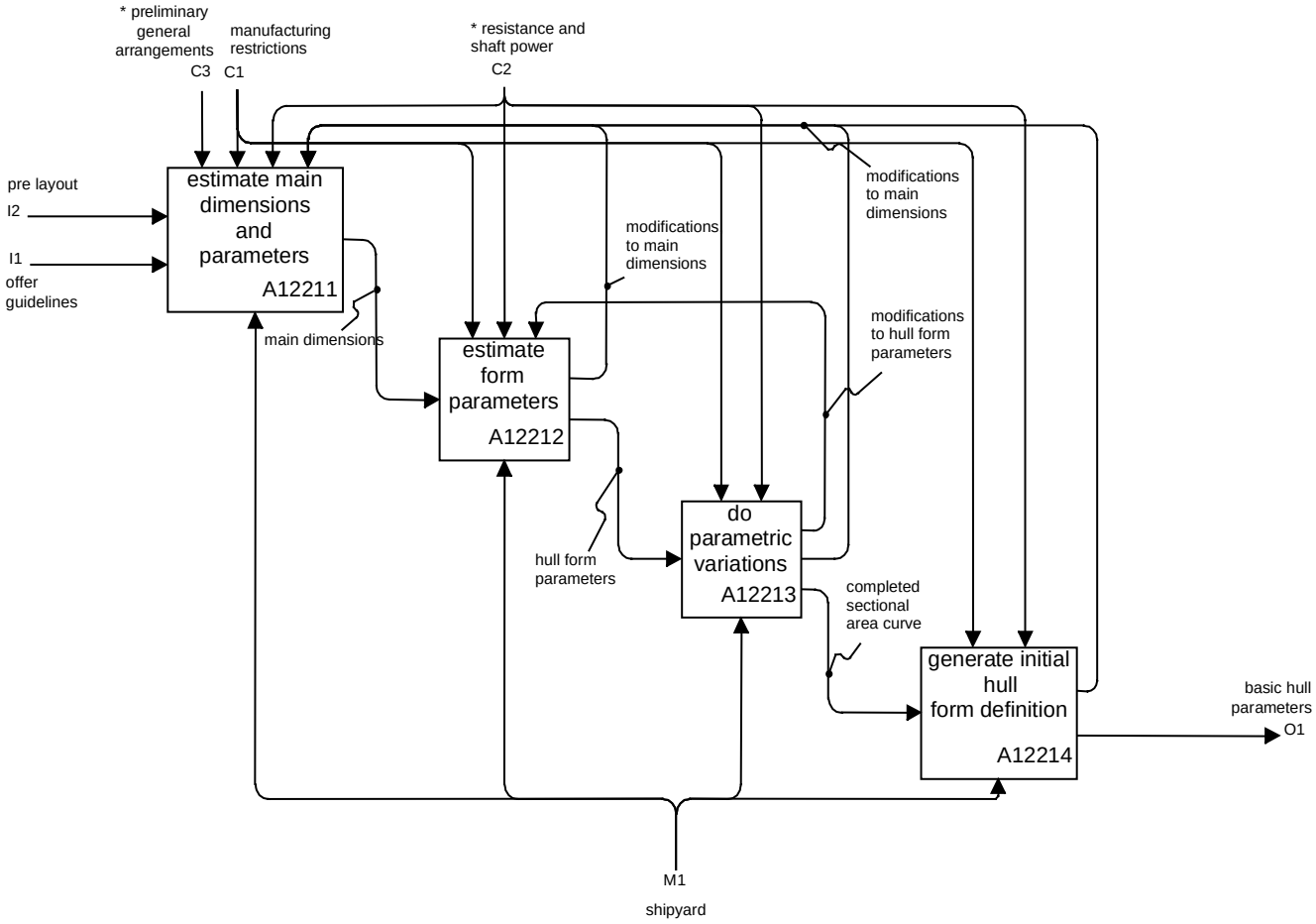


Figure 59 - Node A1221 create preliminary hull form

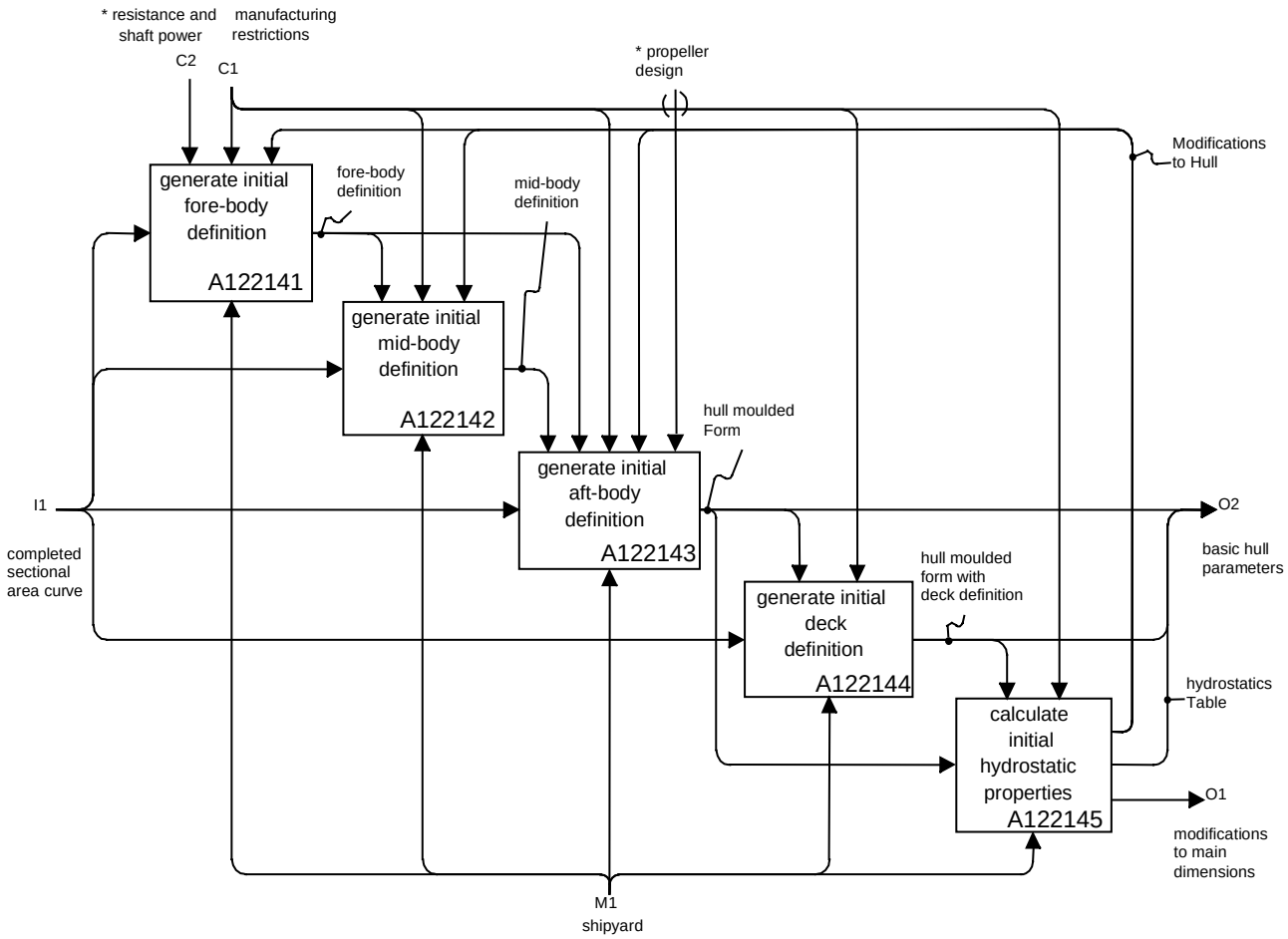


Figure 60 - Node A12214 generate initial hull form definition

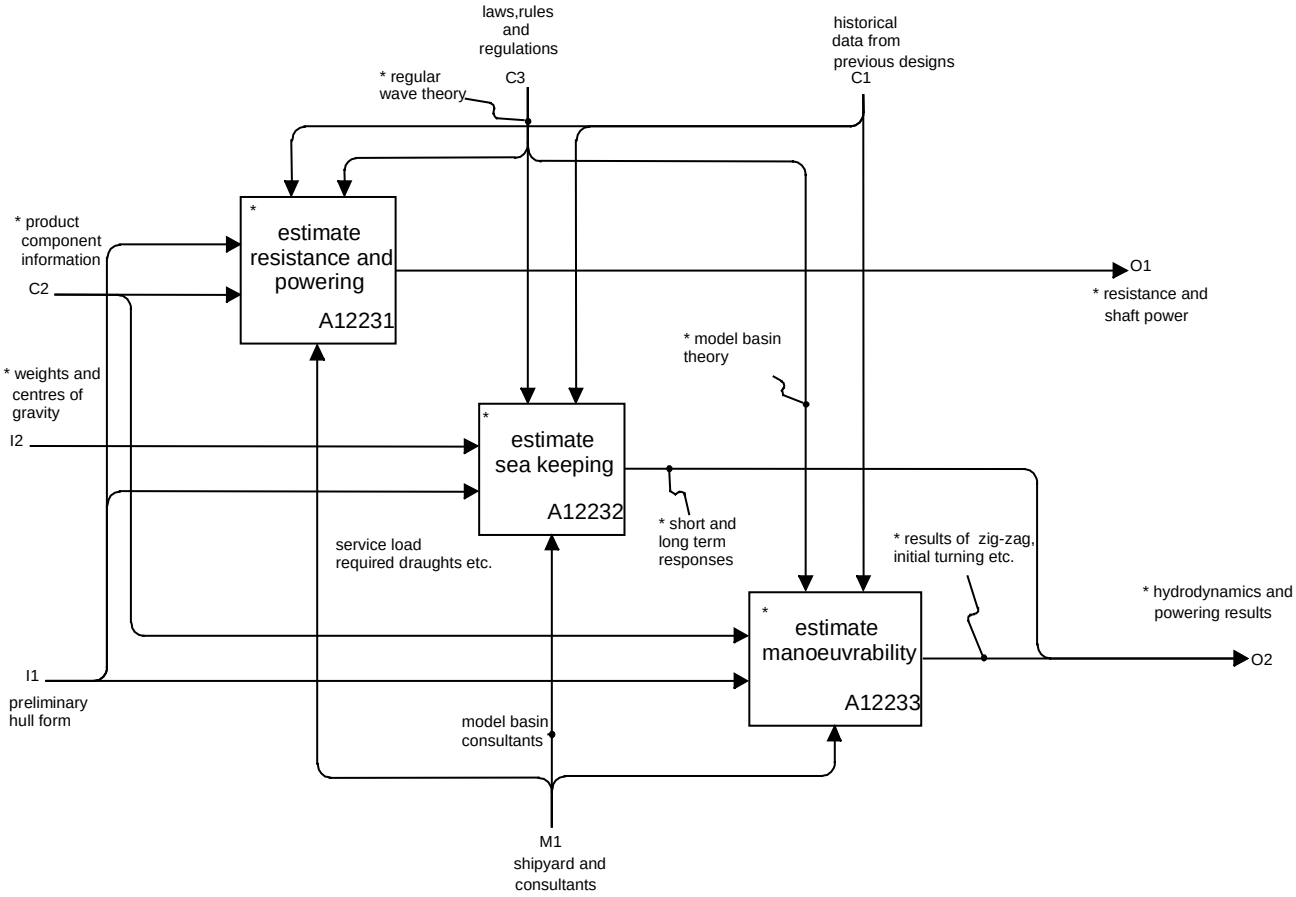


Figure 61 - Node A1223 estimate hydrodynamics and powering

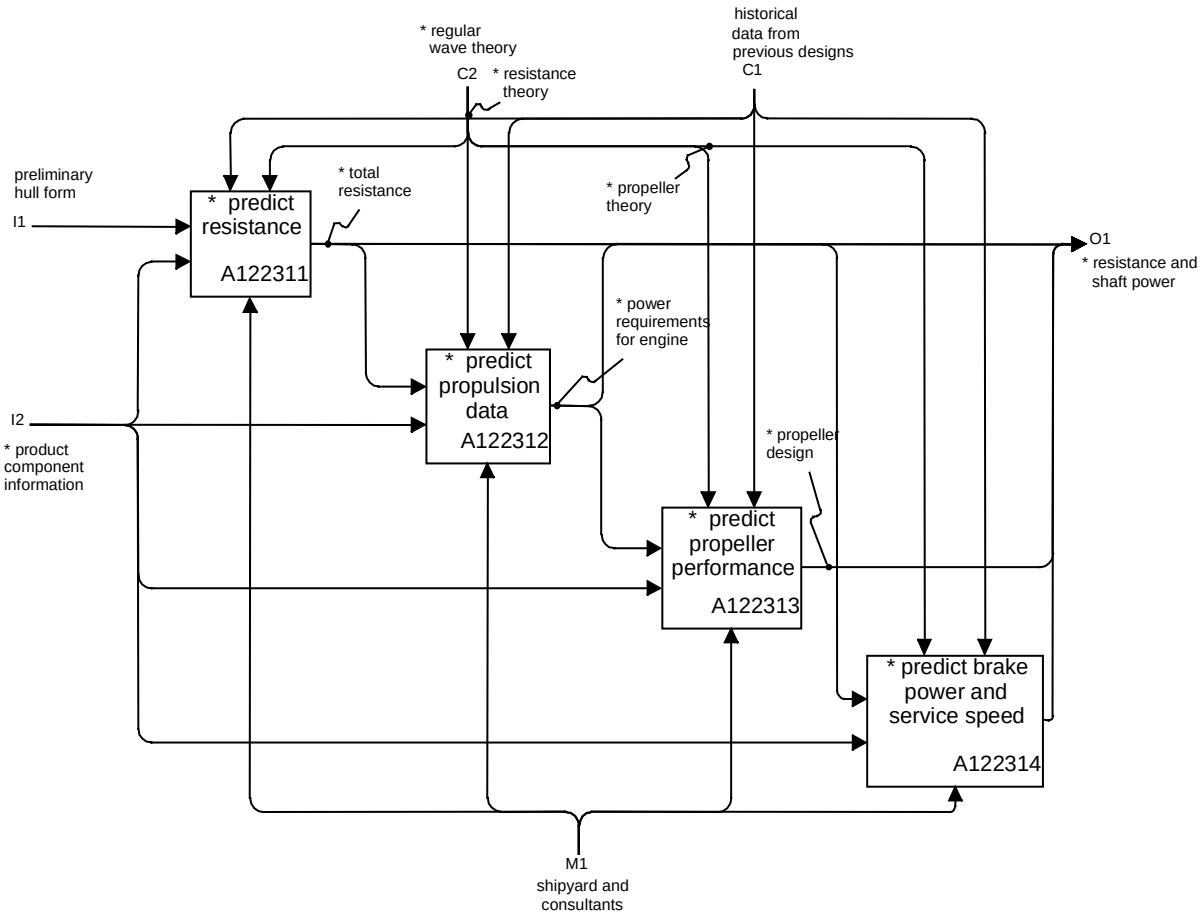


Figure 62 - Node A12231 estimate resistance and powering

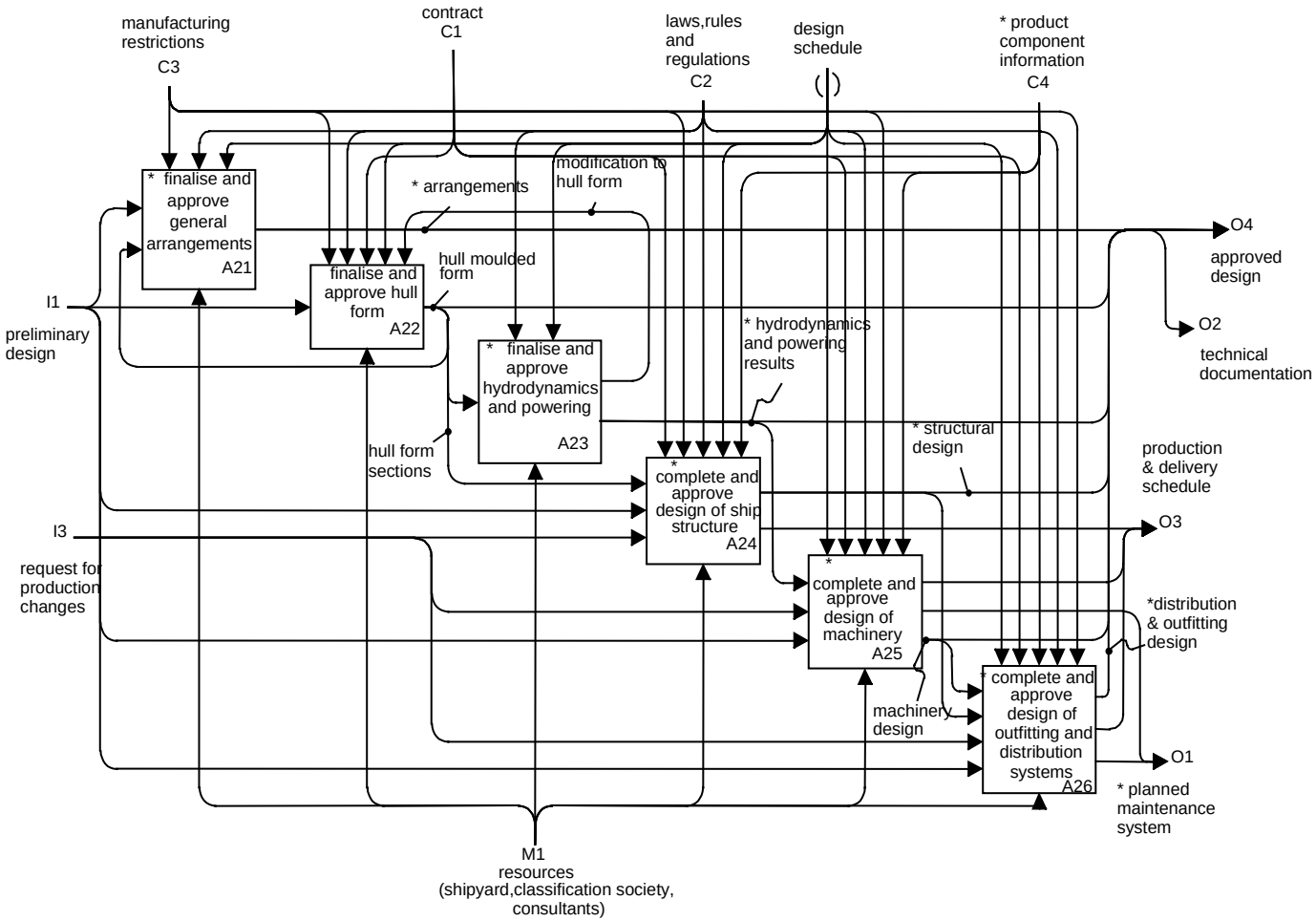


Figure 63 - Node A2 complete and approve ship design

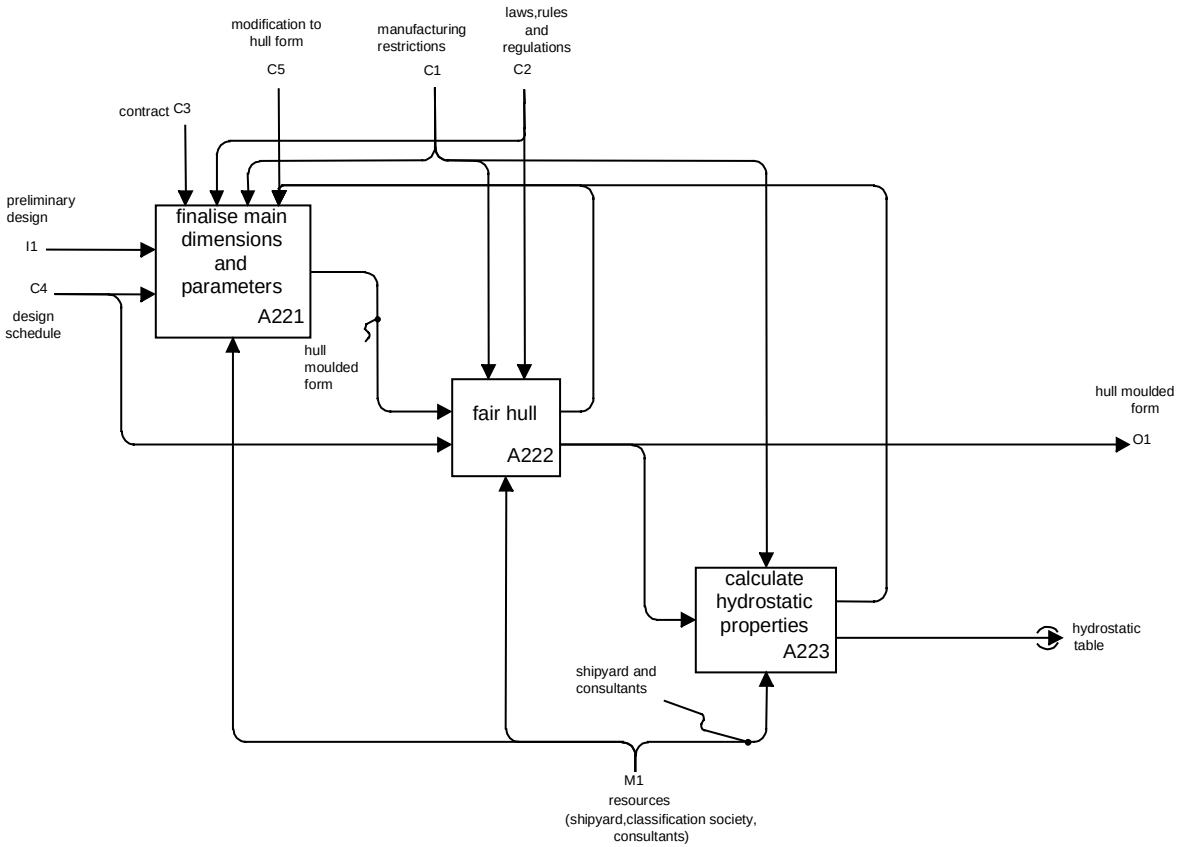


Figure 64 - Node A22 finalise and approve hull form

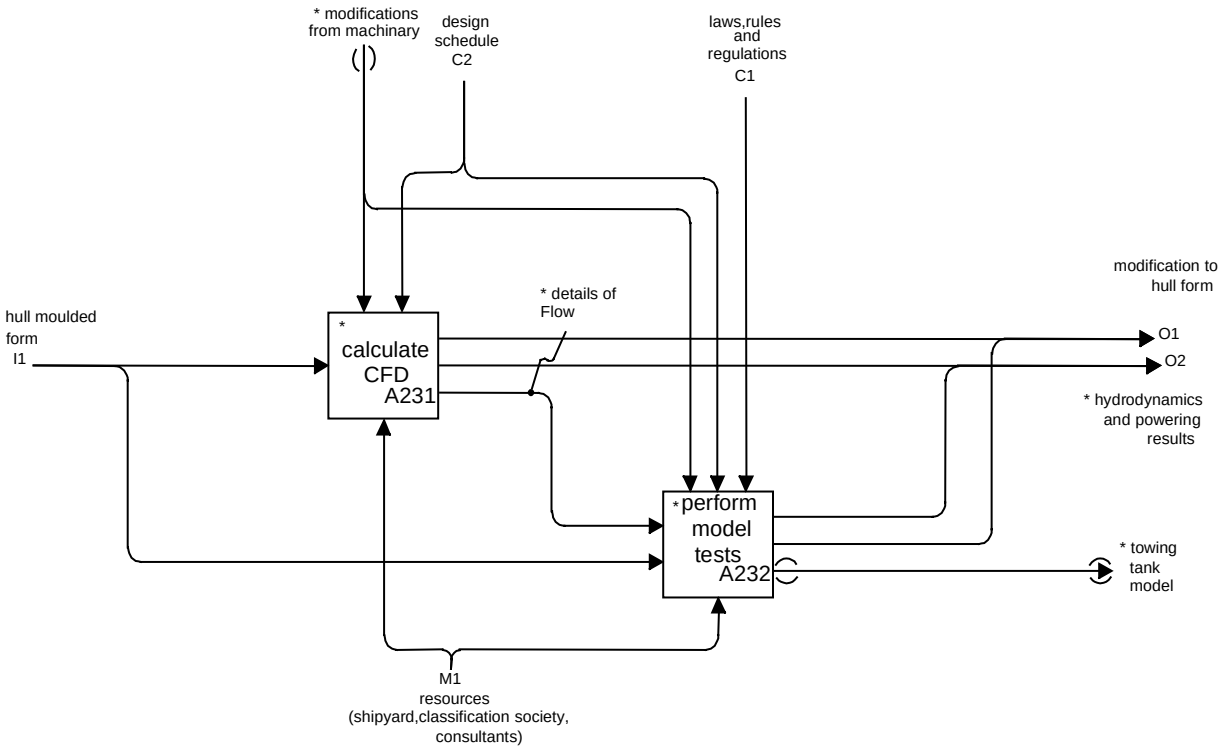


Figure 65 - Node A23 finalise and approve hydrodynamics and powering

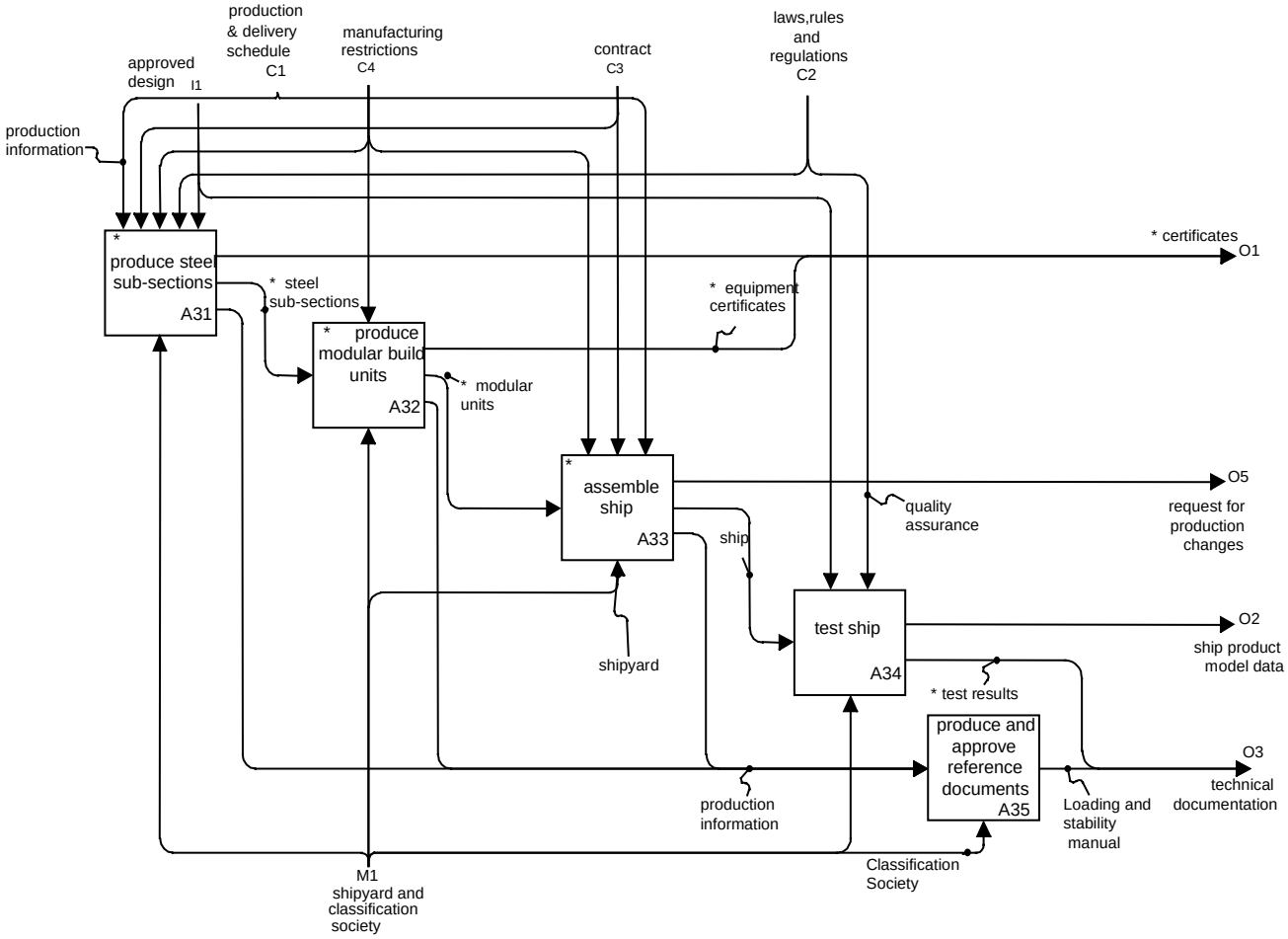


Figure 66 - Node A3 produce and inspect a ship

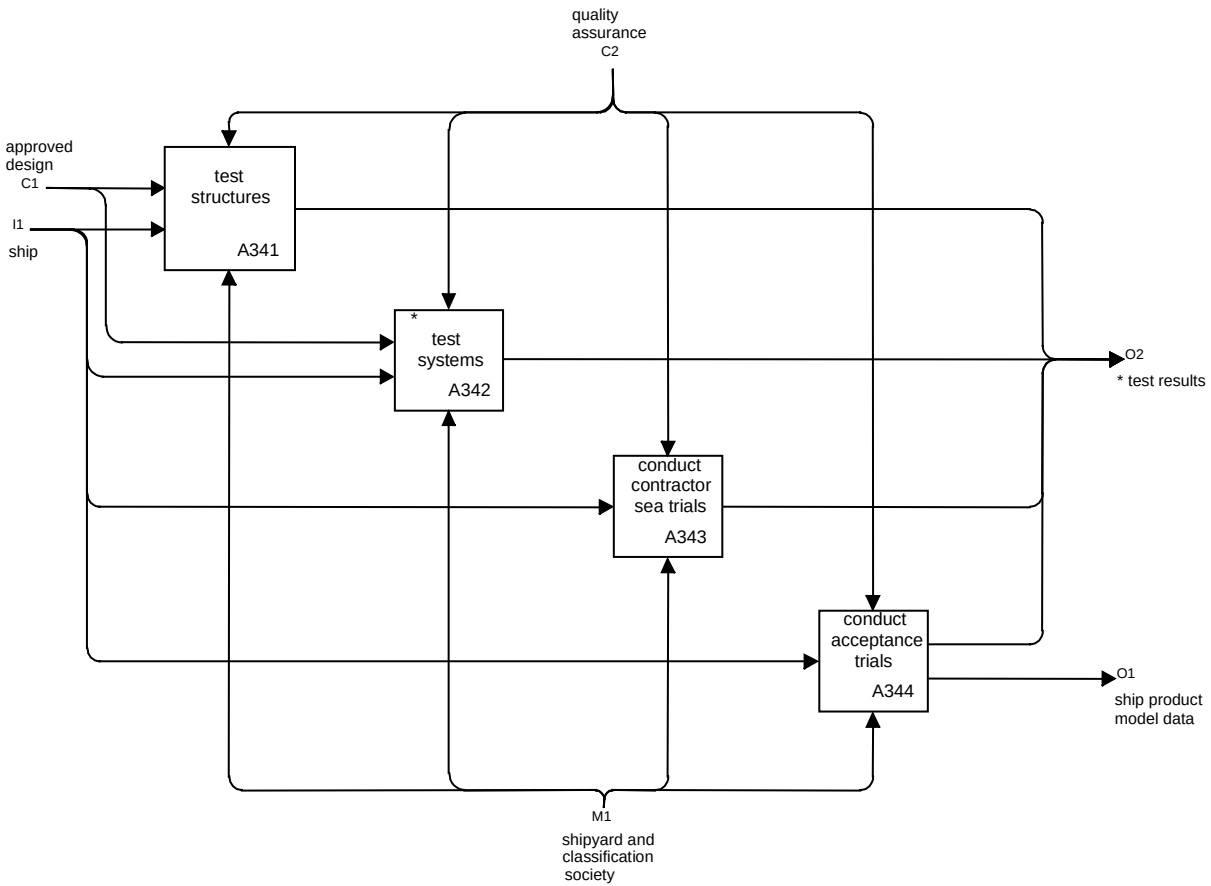


Figure 67 - Node A34 test ship

Annex G
(informative)

Application reference model

This annex provides the application reference model for this part of ISO 10303 and is given in Figure 69 to Figure 101. The application reference model is a graphical representation of the structure and constraints of the application objects specified in clause 4. The graphical form of the application reference model is presented in EXPRESS-G. The application reference model is independent of any implementation method.

EXPRESS-G is defined in Annex D of ISO 10303-11.

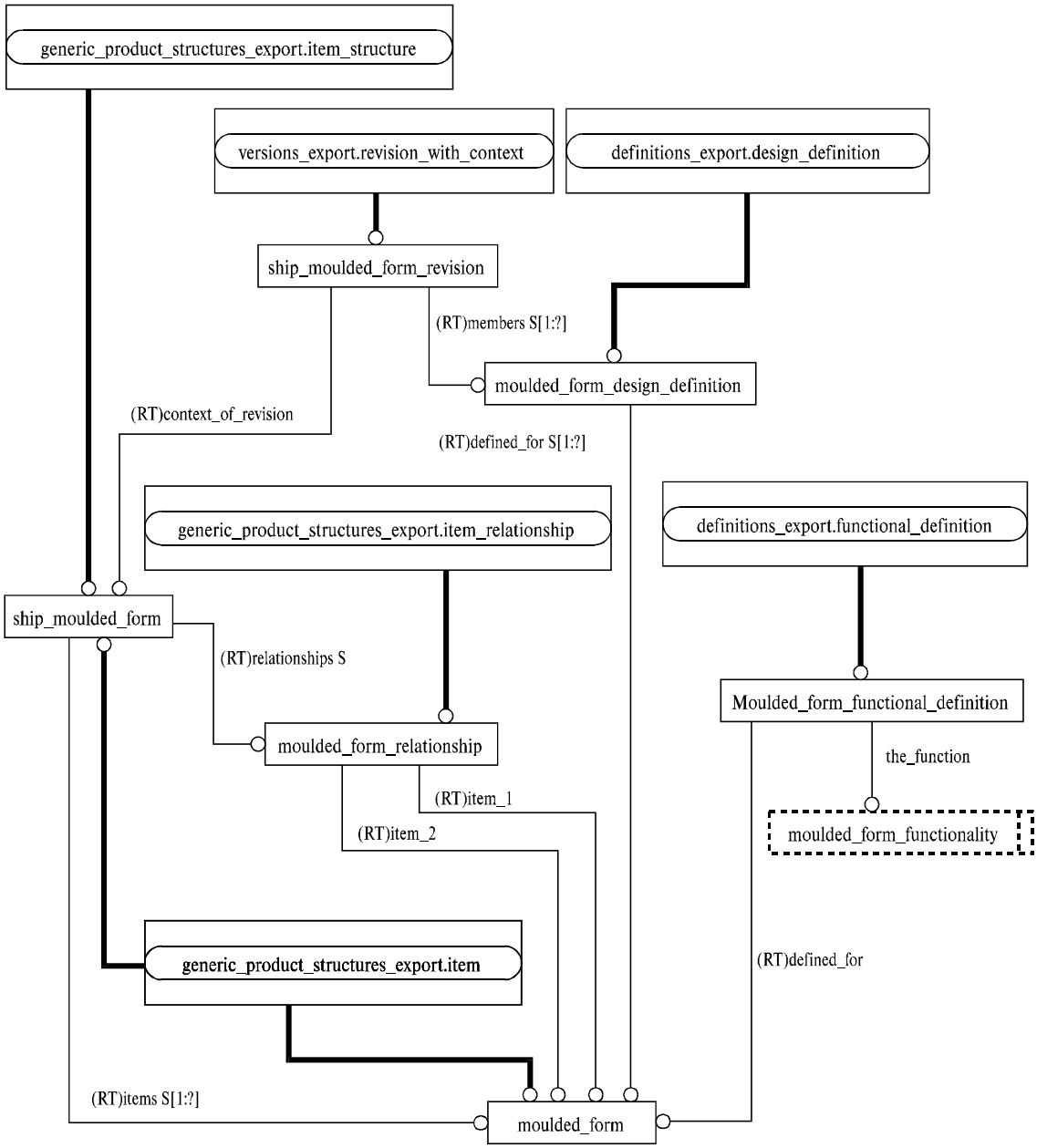


Figure 68 - ARM EXPRESS-G UoF Ship moulded forms diagram 1

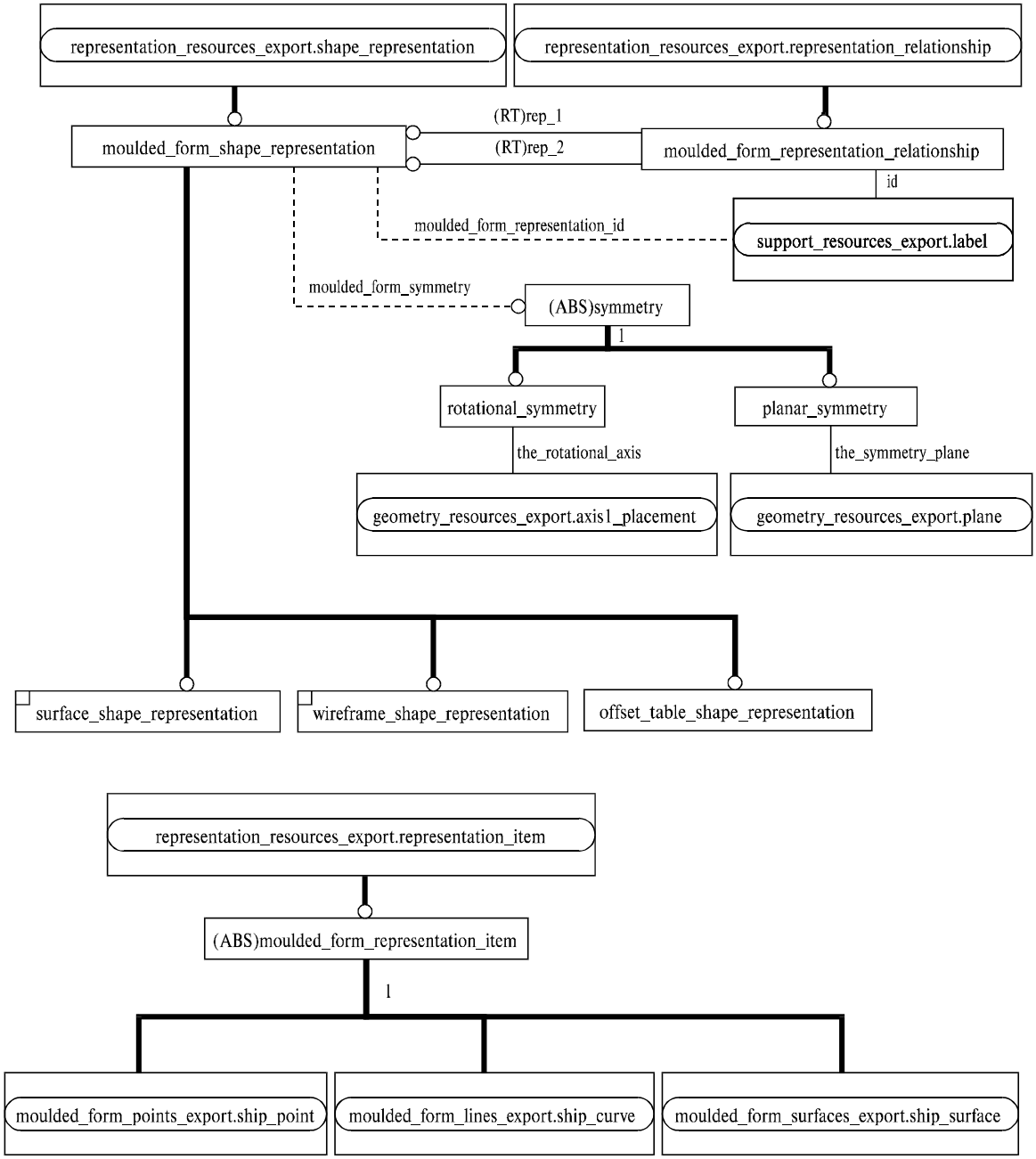


Figure 69 - ARM EXPRESS-G UoF Ship moulded forms diagram 2

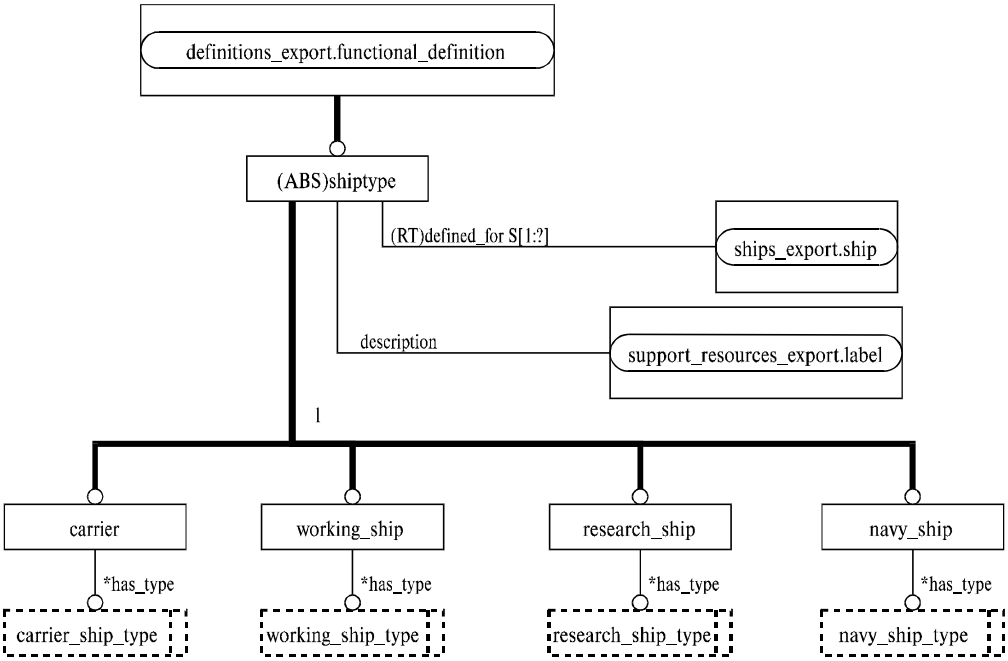


Figure 70 - ARM EXPRESS-G UoF Ship general characteristics diagram 1

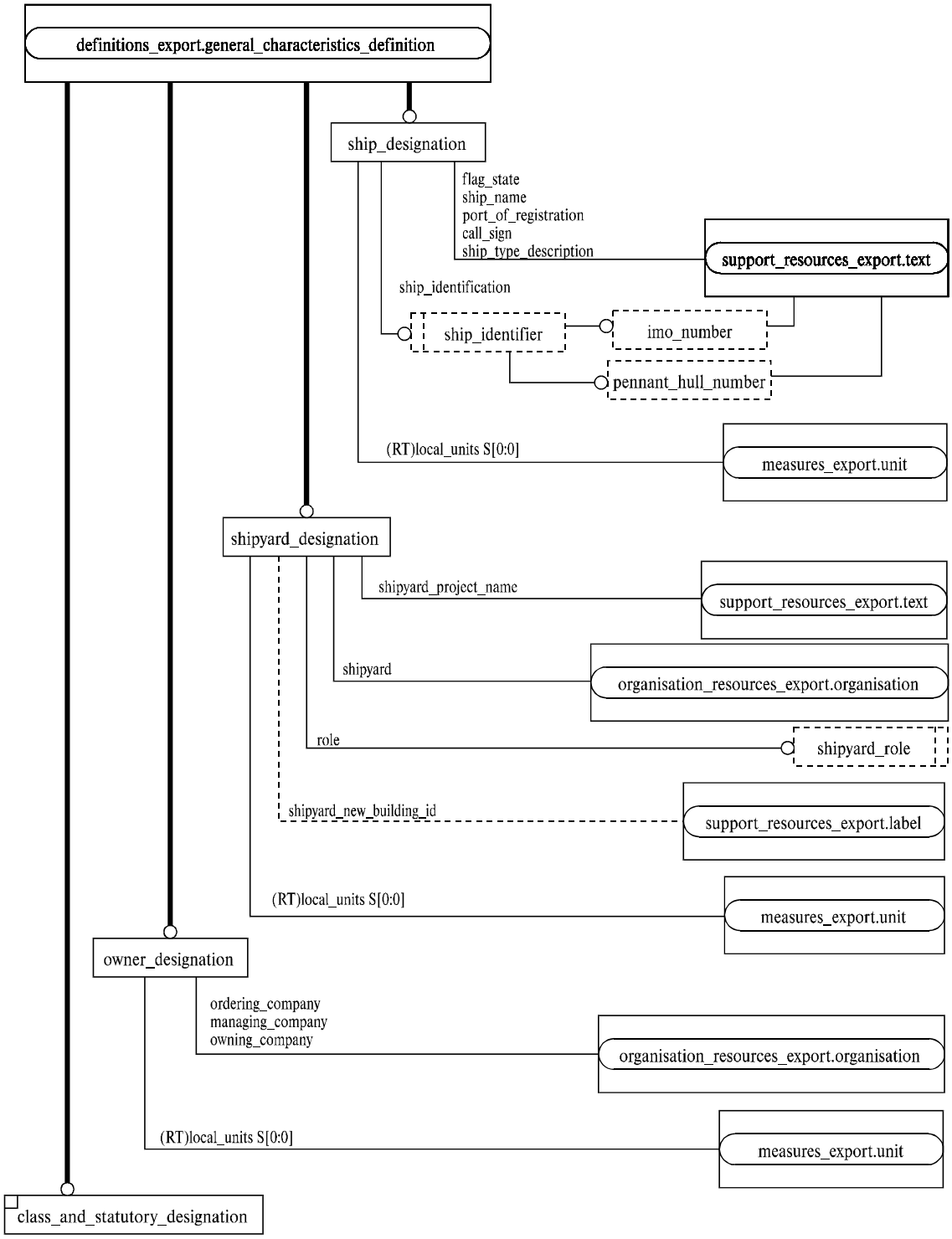


Figure 71 - ARM EXPRESS-G UoF Ship general characteristics diagram 2

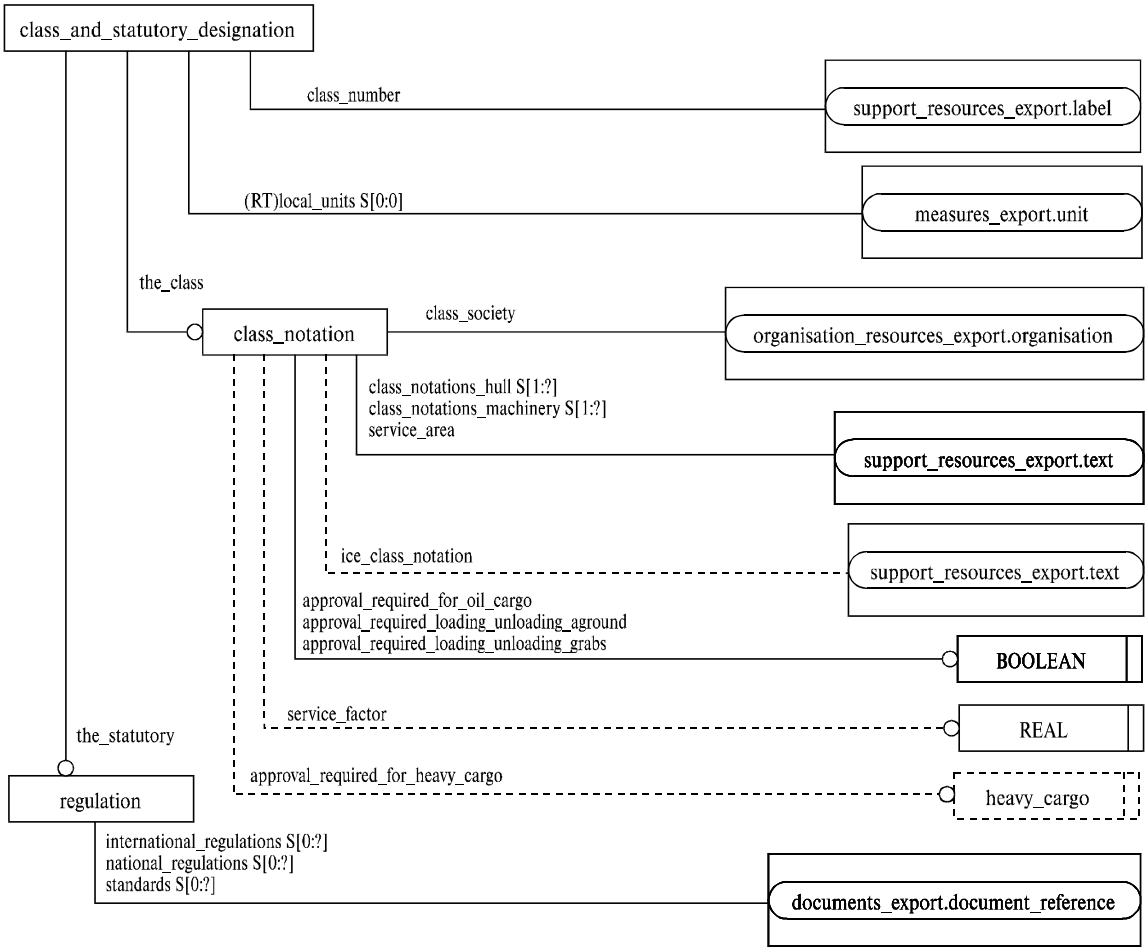


Figure 72 - ARM EXPRESS-G UoF Ship general characteristics diagram 3

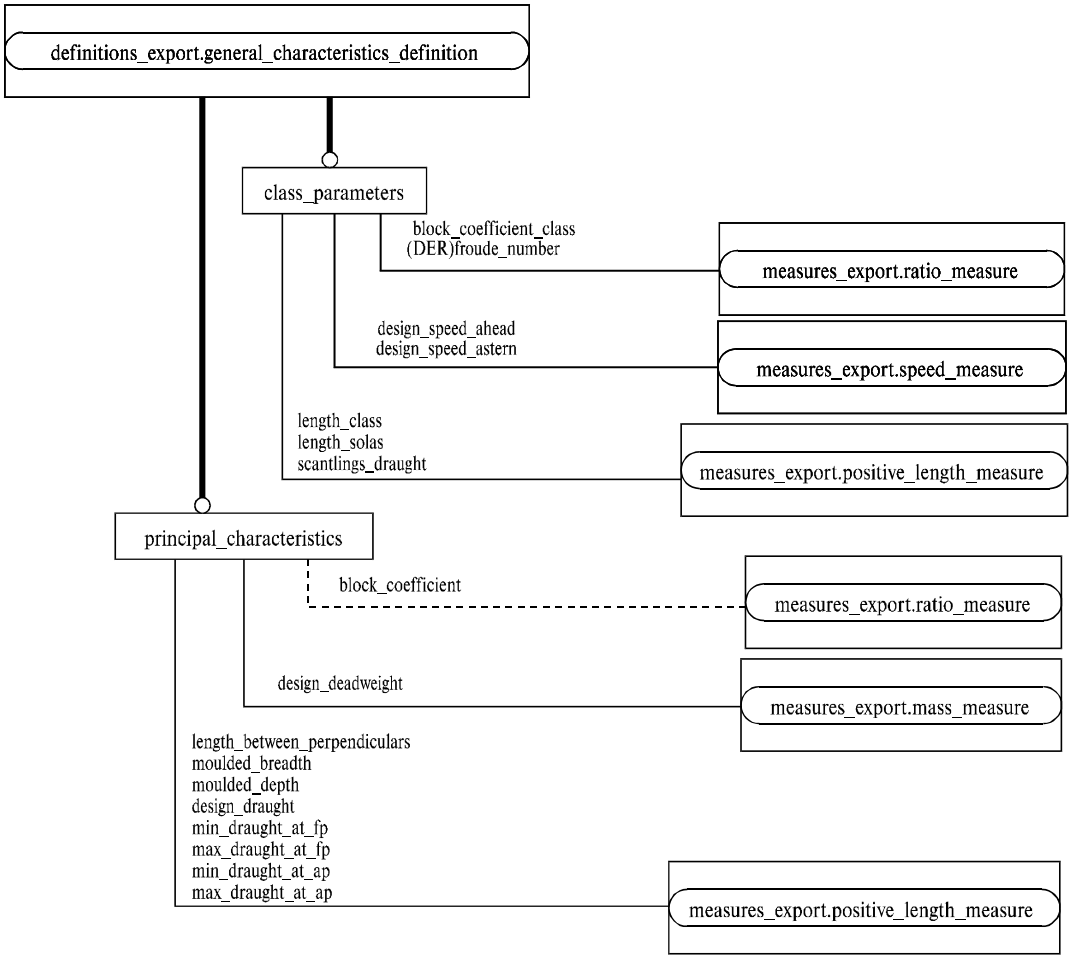


Figure 73 - ARM EXPRESS-G UoF Ship general characteristics diagram 4

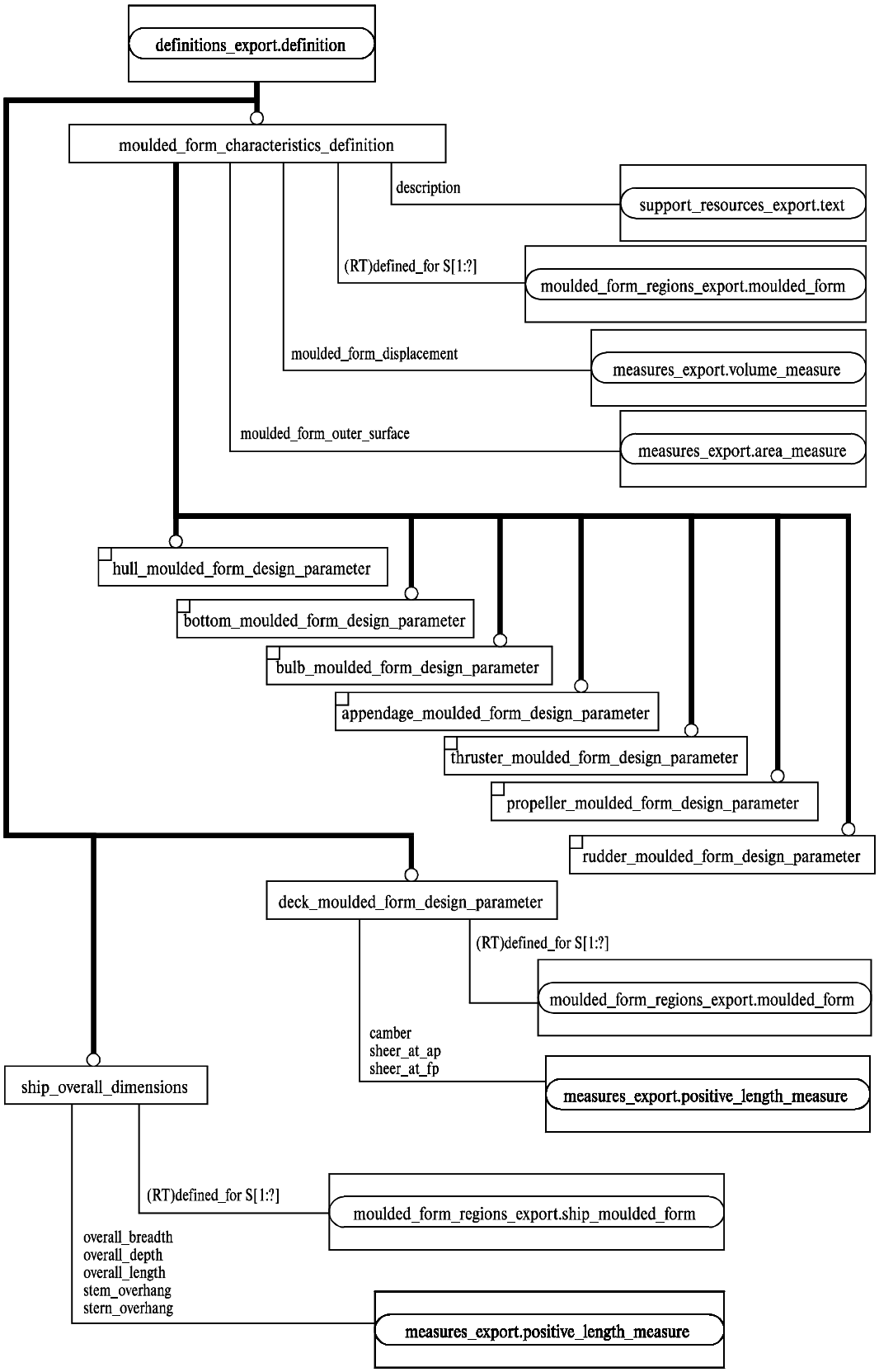


Figure 74 - ARM EXPRESS-G UoF Ship design parameter diagram 1

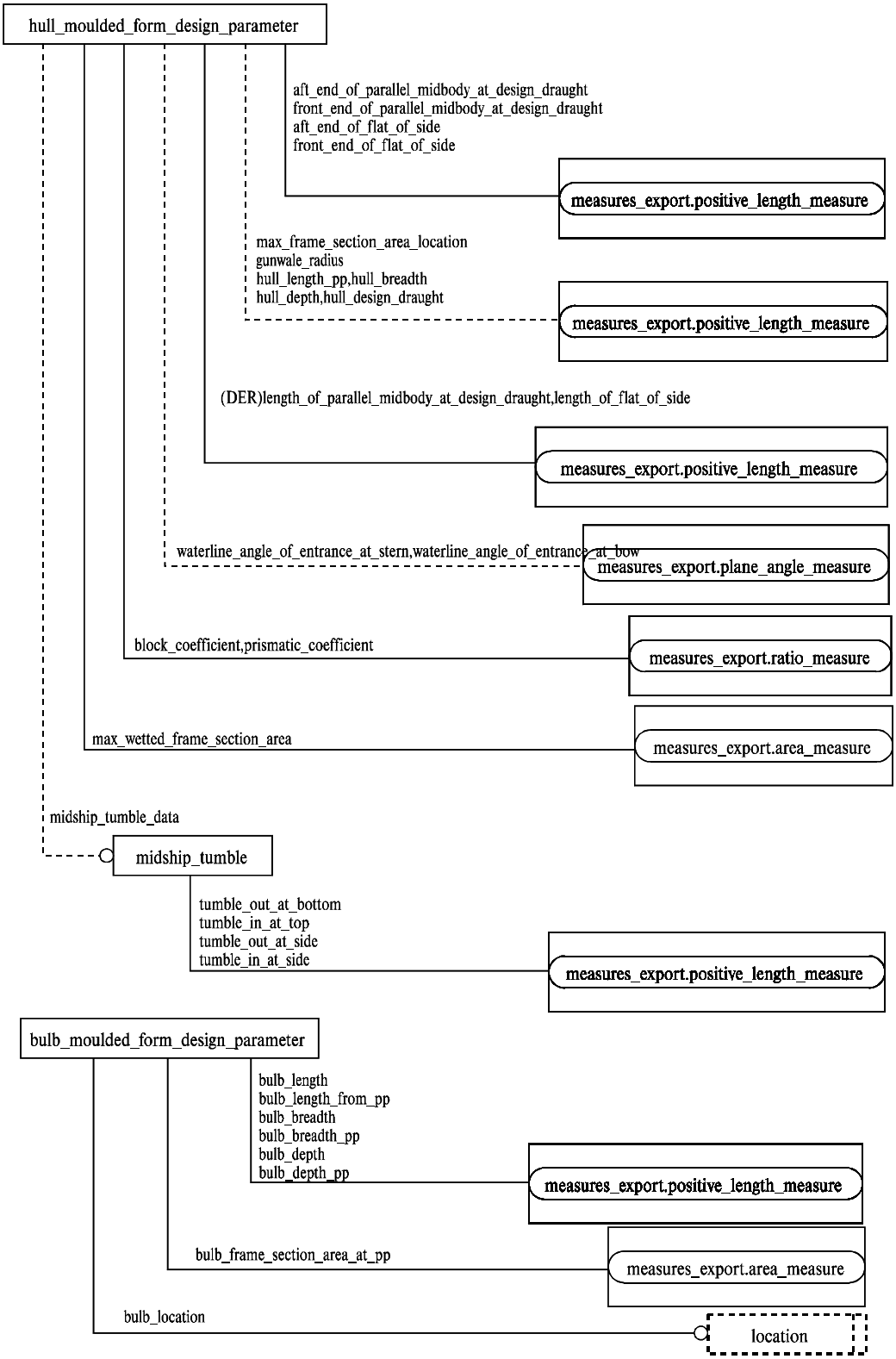


Figure 75 - ARM EXPRESS-G UoF Ship design parameter diagram 2

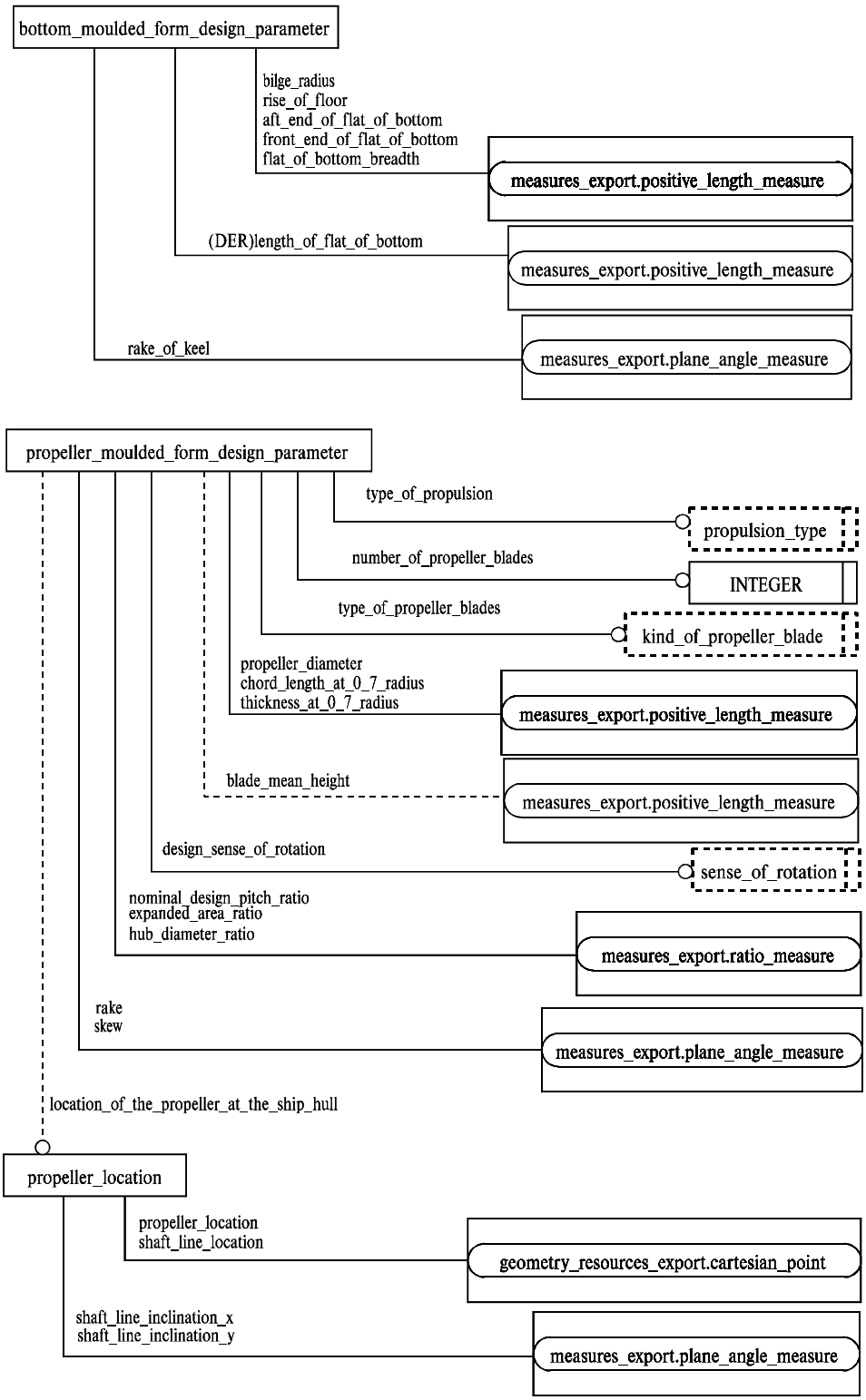


Figure 76 - ARM EXPRESS-G UoF Ship design parameter diagram 3

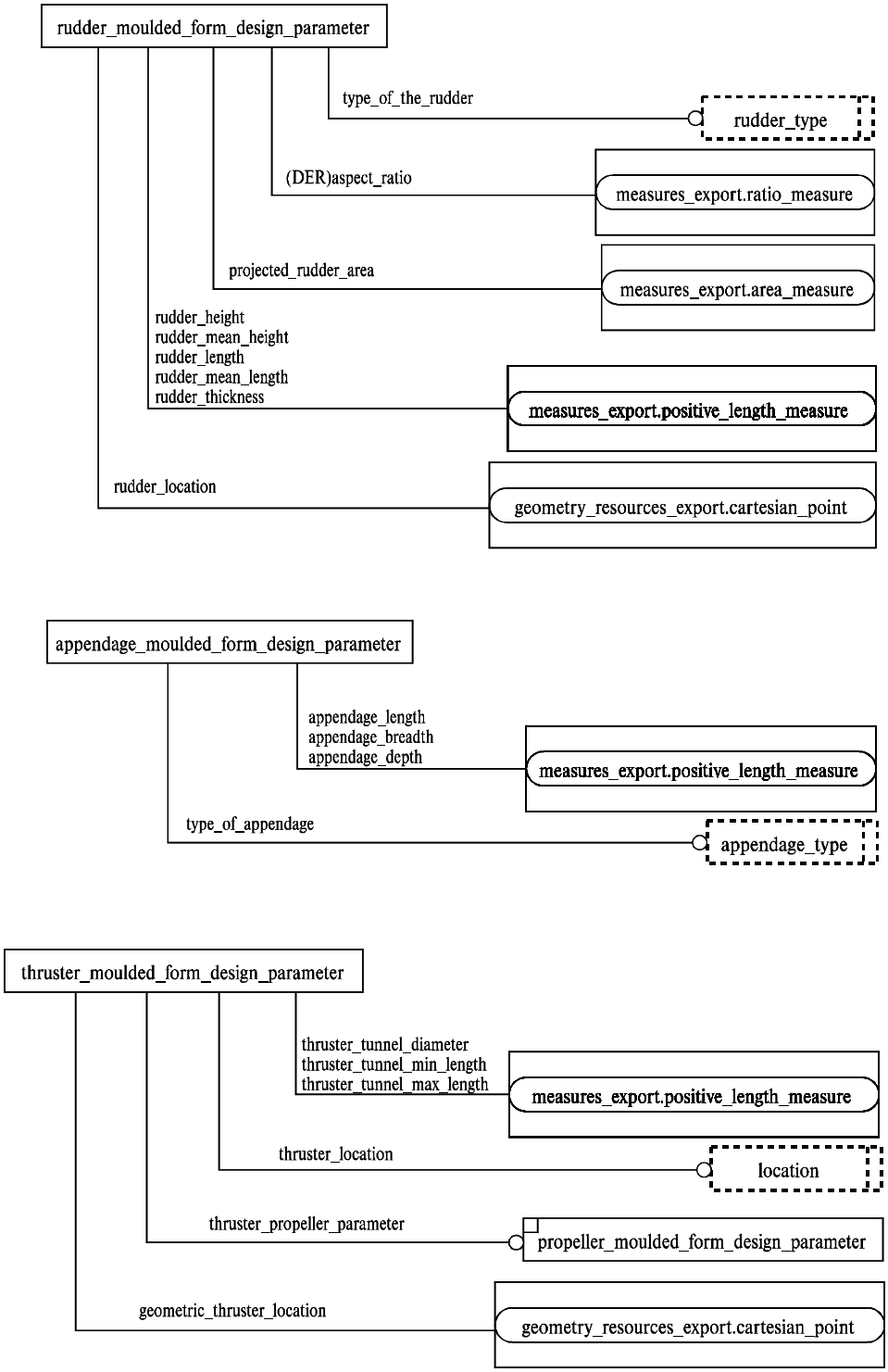


Figure 77 - ARM EXPRESS-G UoF Ship design parameter diagram 4

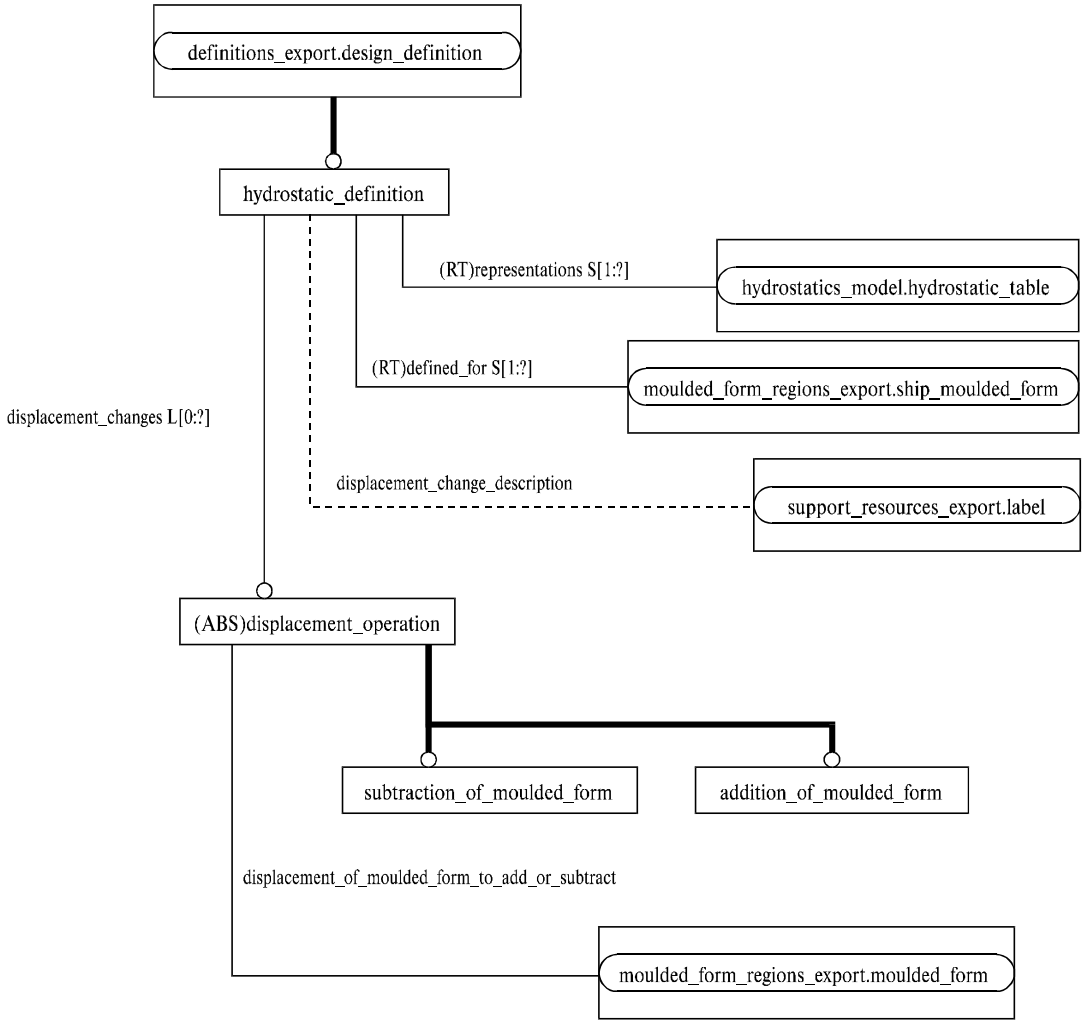


Figure 78 - ARM EXPRESS-G UoF Hydrostatics diagram 1

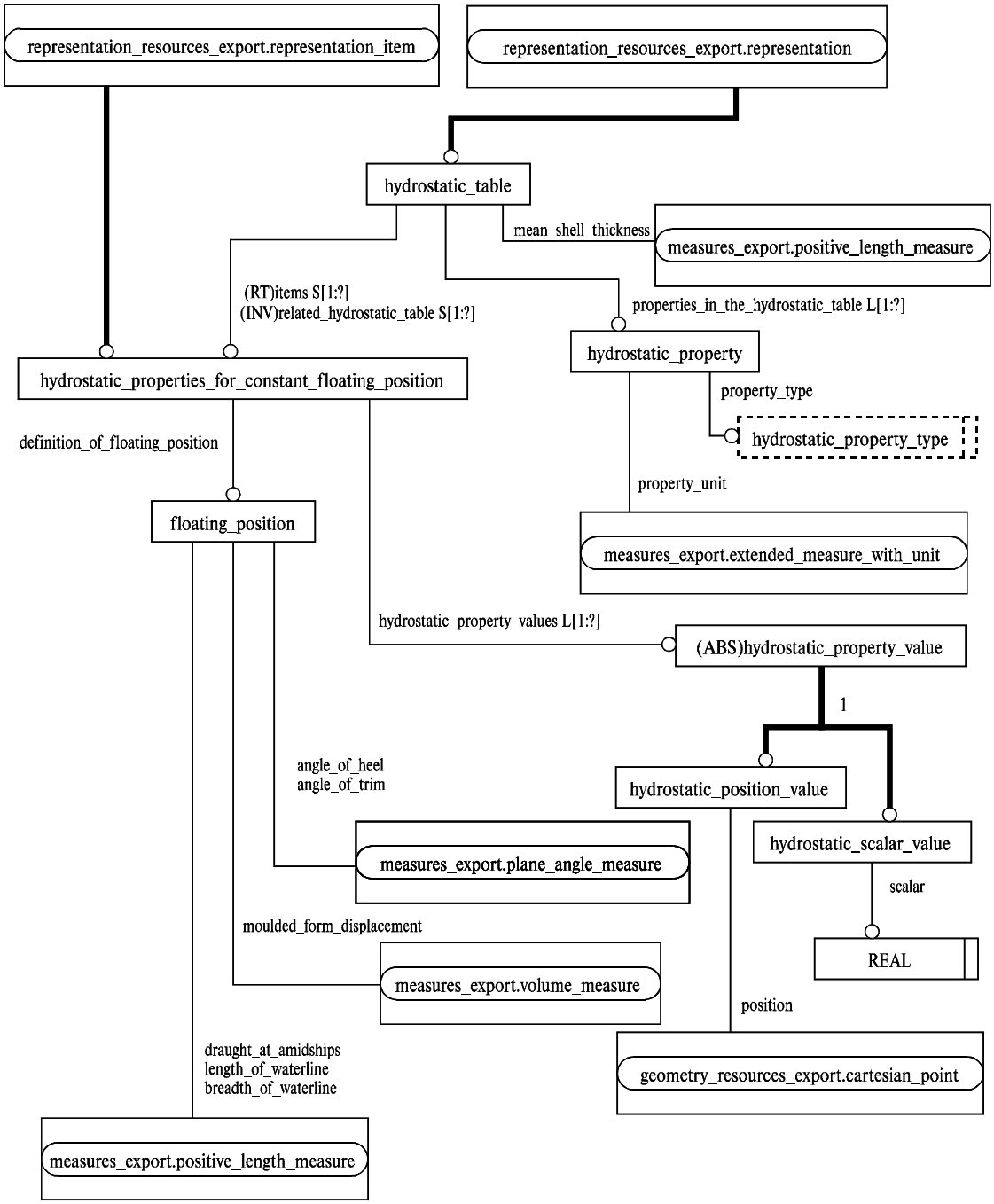


Figure 79 - ARM EXPRESS-G UoF Hydrostatics diagram 2

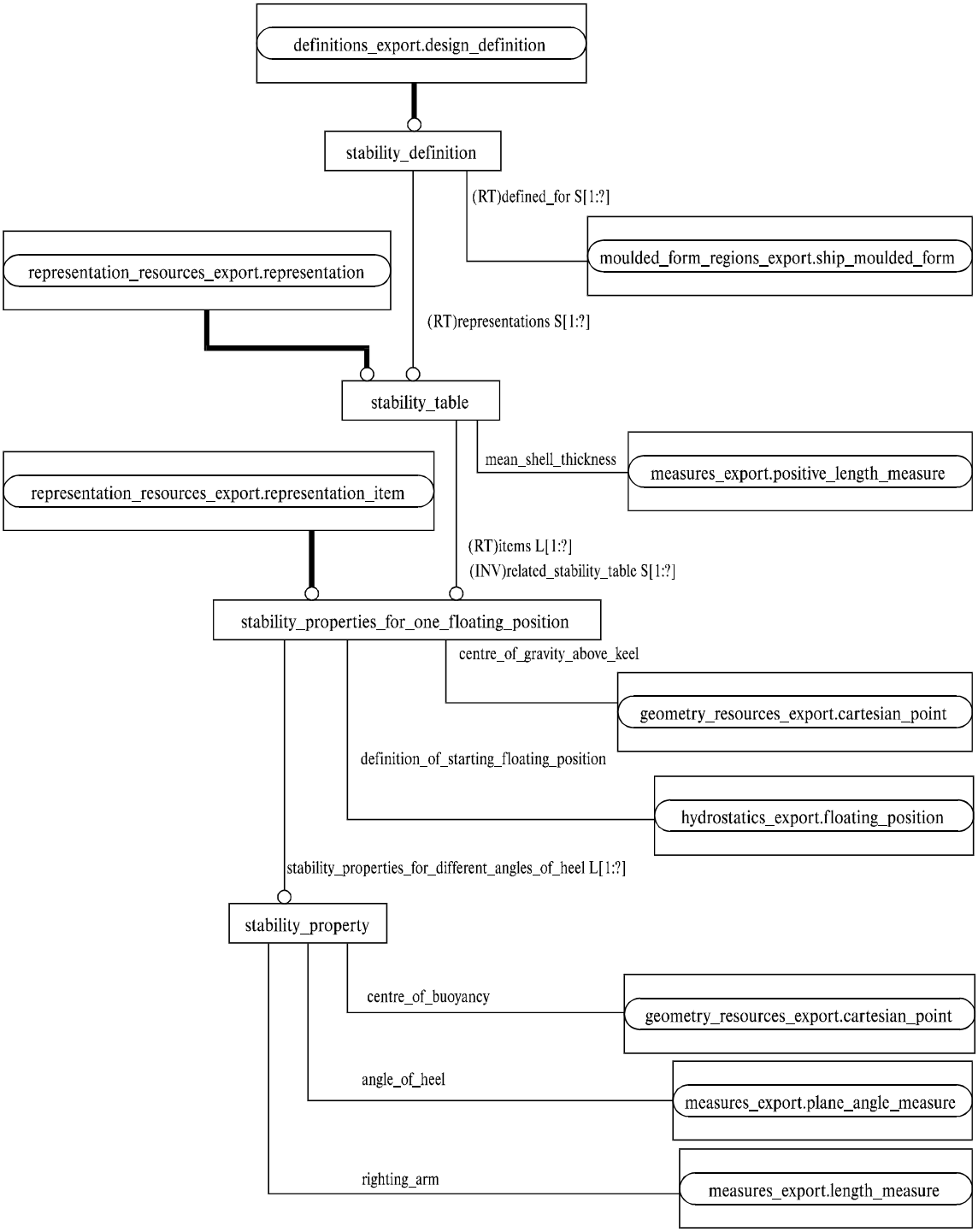


Figure 80 - ARM EXPRESS-G UoF Hydrostatics diagram 3

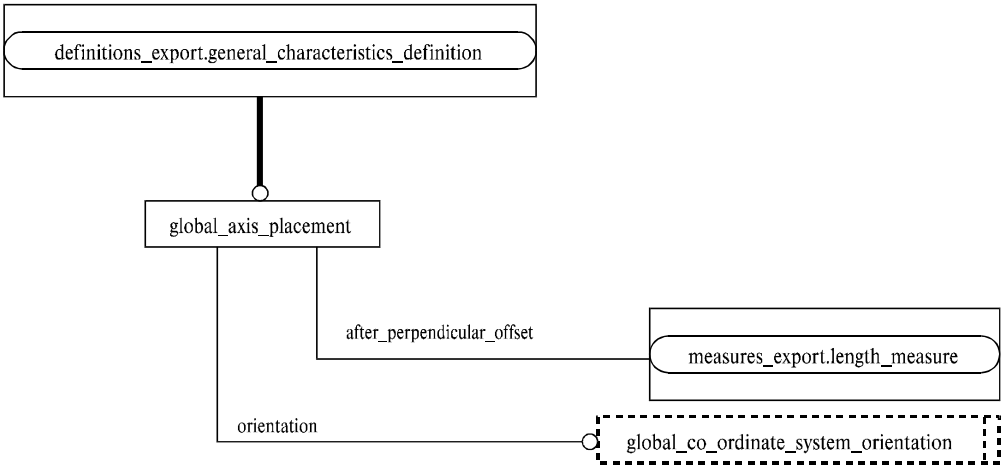


Figure 81 - ARM EXPRESS-G UoF Location concepts diagram 1

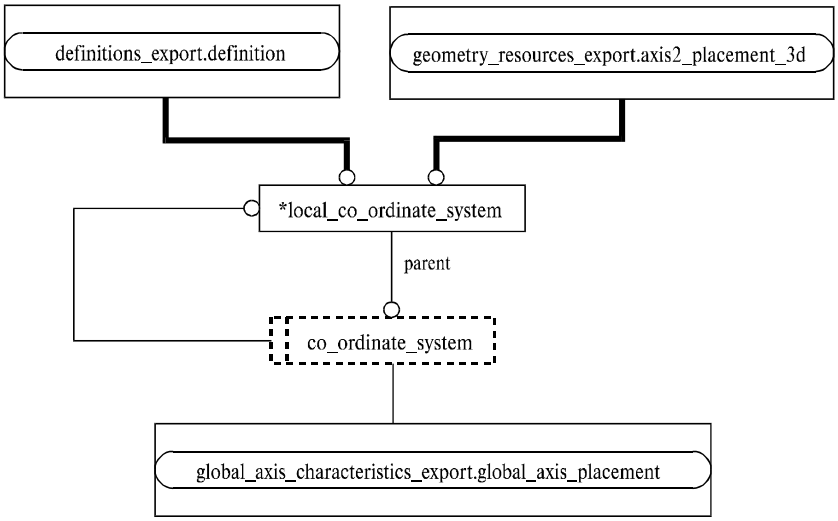


Figure 82 - ARM EXPRESS-G UoF Location concepts diagram 2

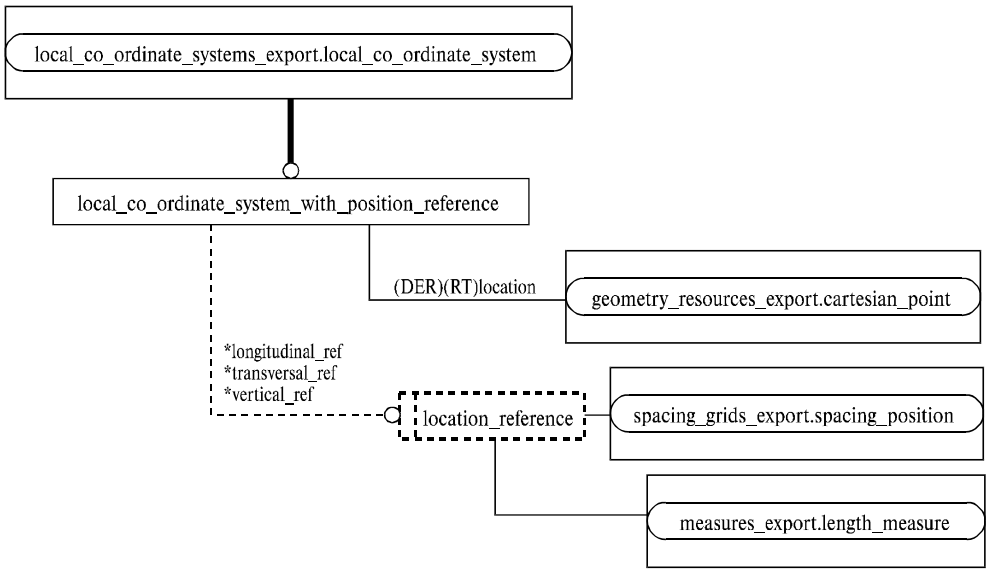


Figure 83 - ARM EXPRESS-G UoF Location concepts diagram 3

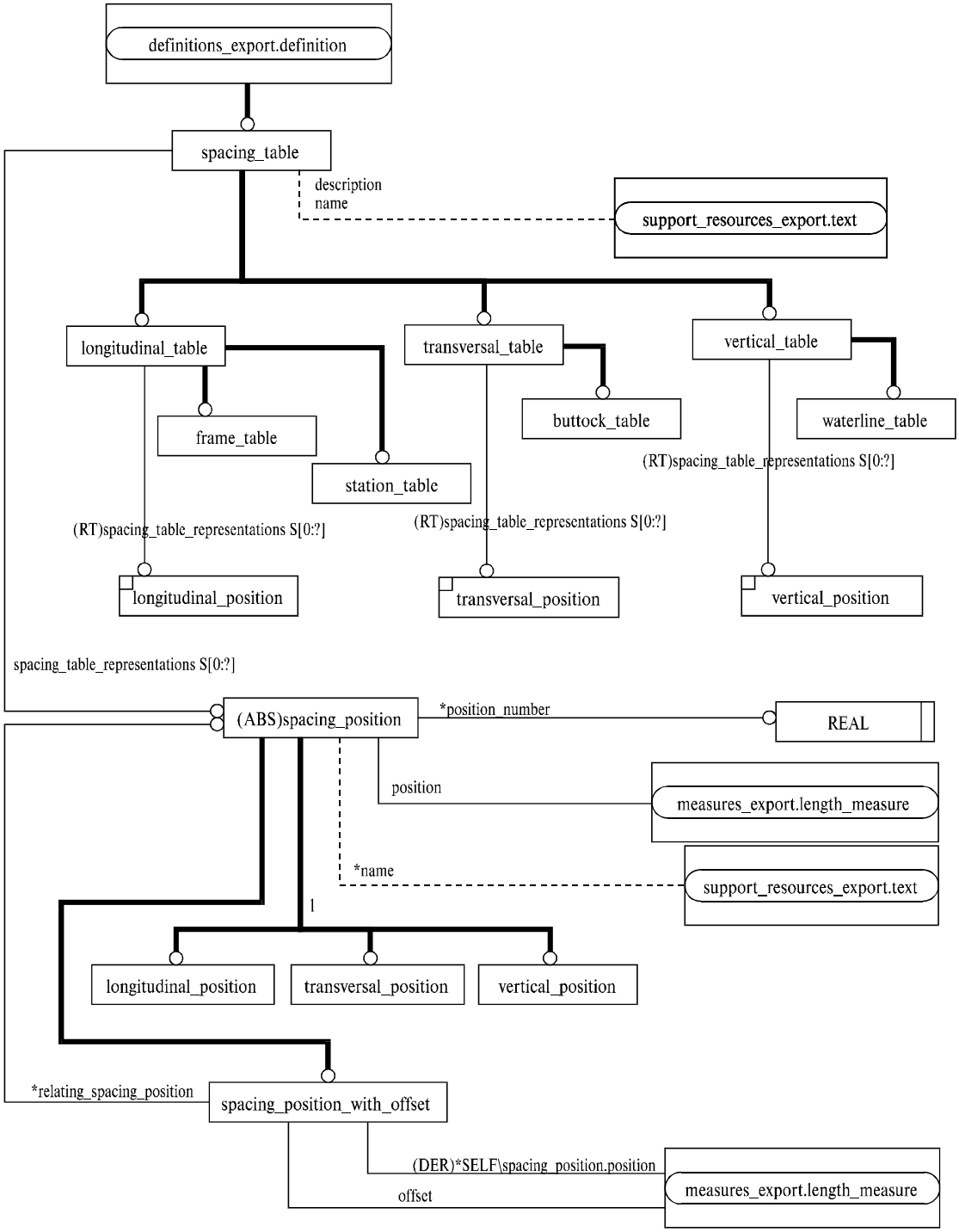


Figure 84 - ARM EXPRESS-G UoF Location concepts diagram 4

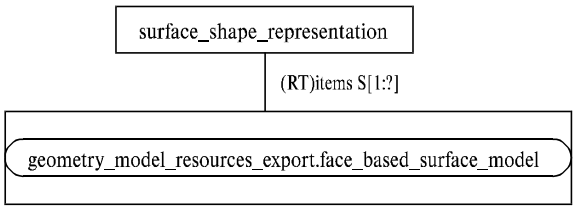


Figure 85 - ARM EXPRESS-G UoF Surface representations

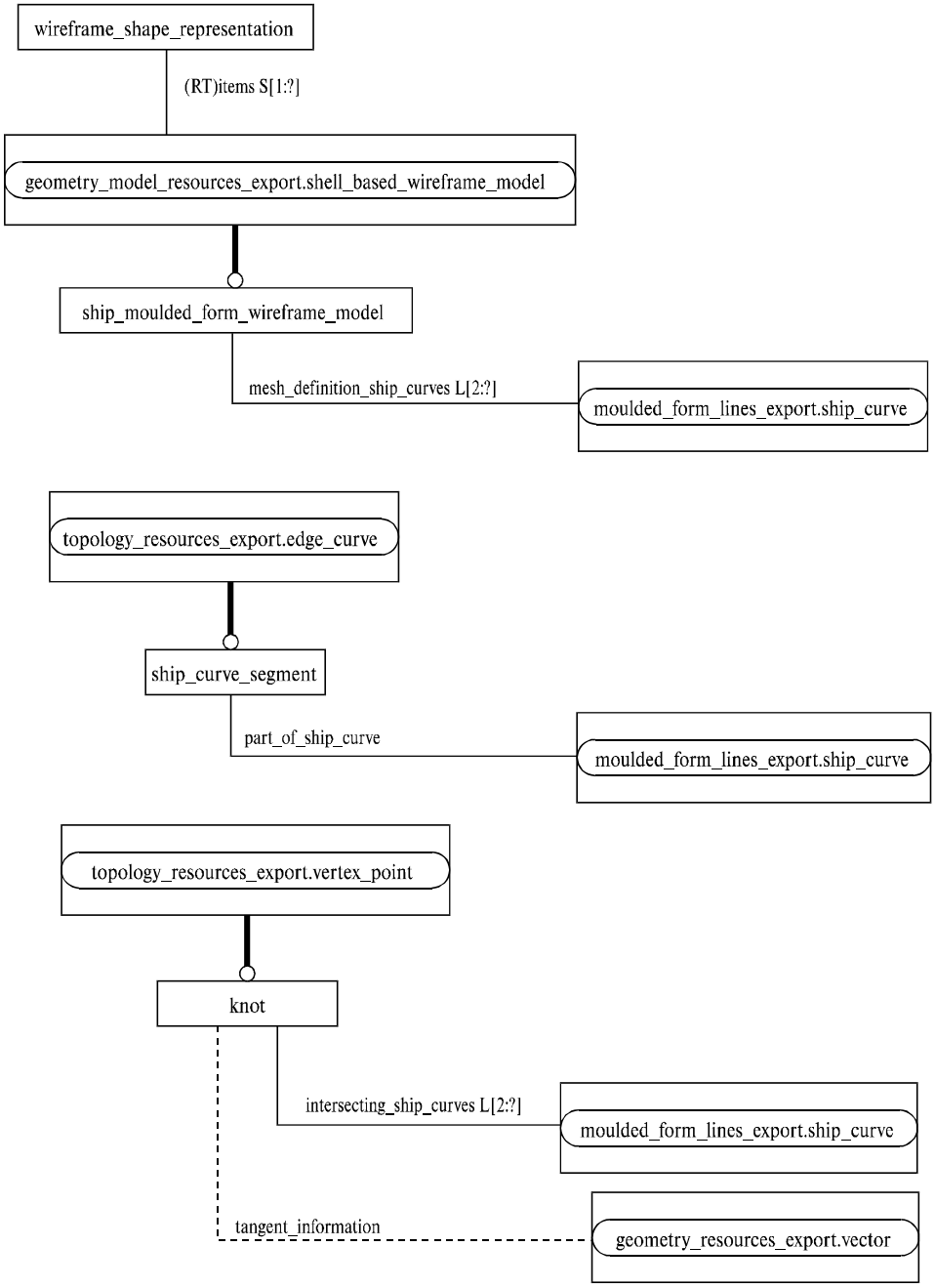


Figure 86 - ARM EXPRESS-G UoF Wireframe representations

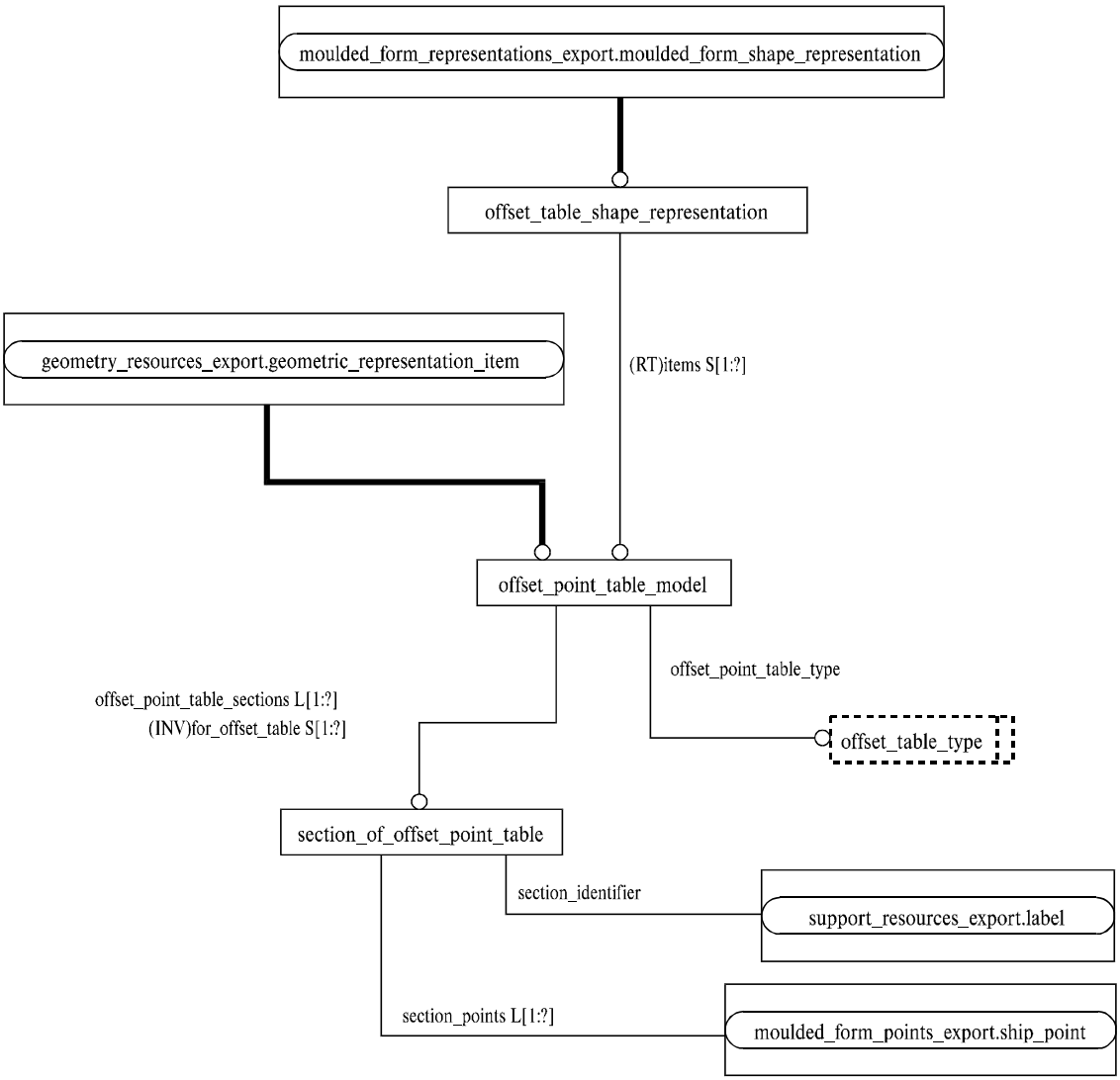


Figure 87 - ARM EXPRESS-G UoF Offset table representations

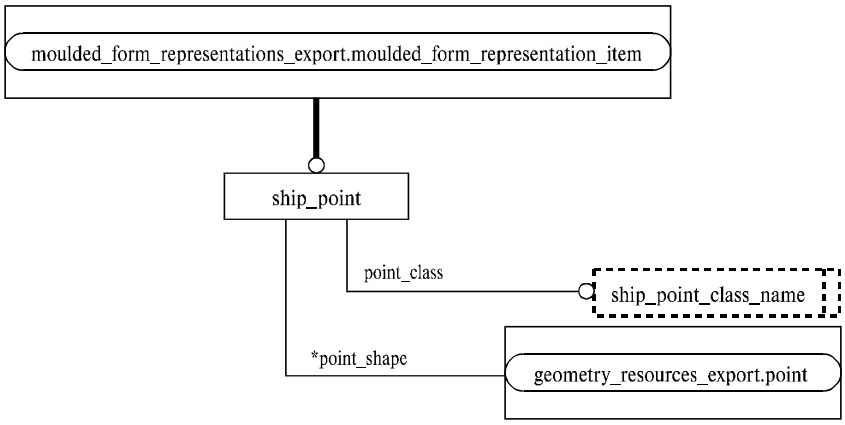


Figure 88 - ARM EXPRESS-G UoF Basic geometry diagram 1

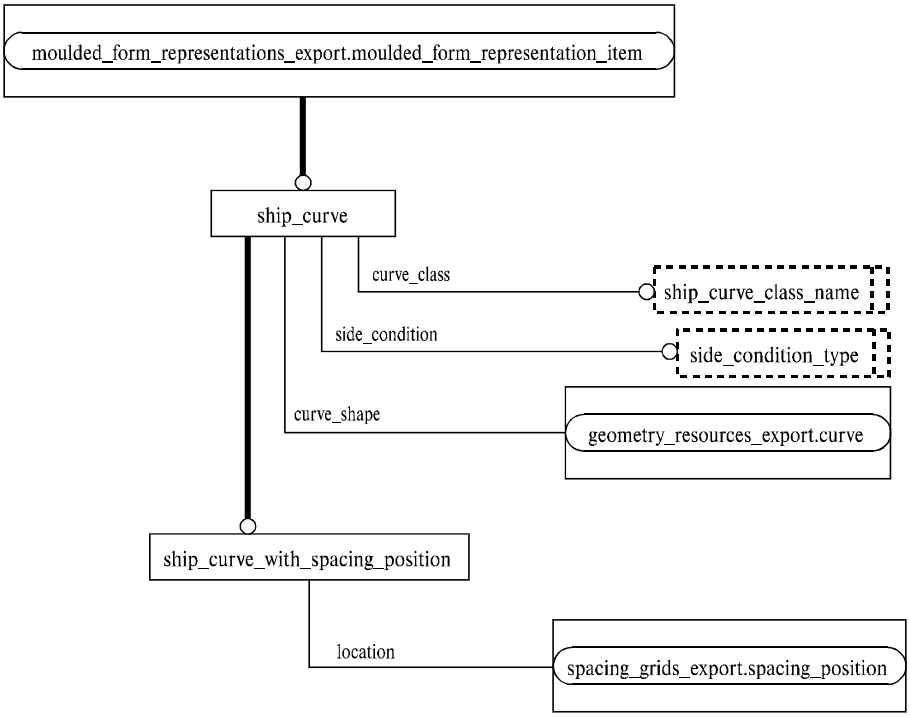


Figure 89 - ARM EXPRESS-G UoF Basic geometry diagram 2

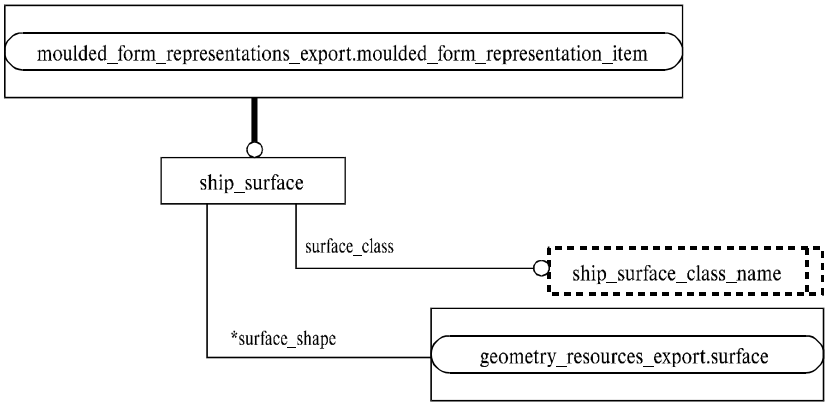


Figure 90 - ARM EXPRESS-G UoF Basic geometry diagram 3

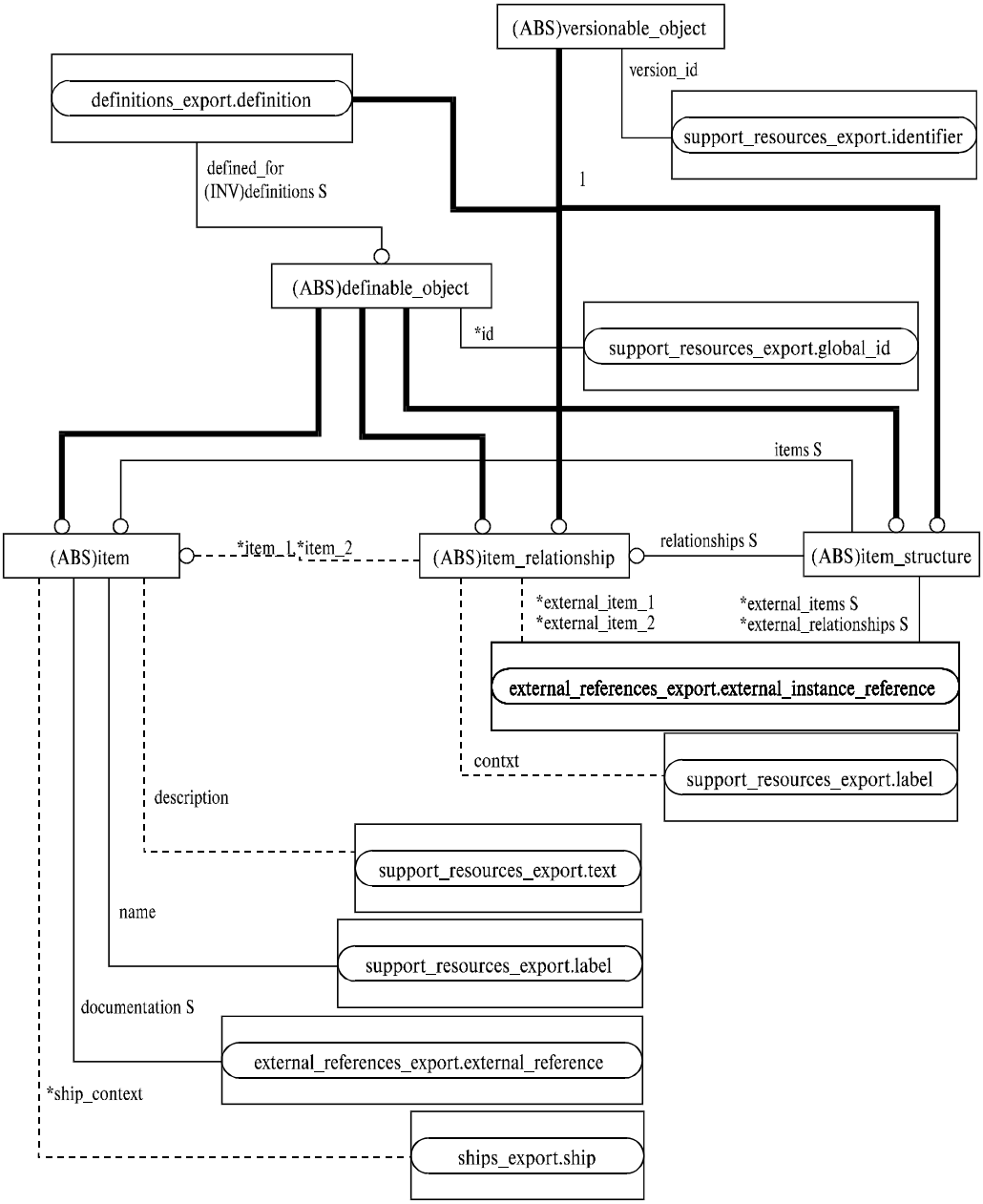


Figure 91 - ARM EXPRESS-G UoF Items diagram 1

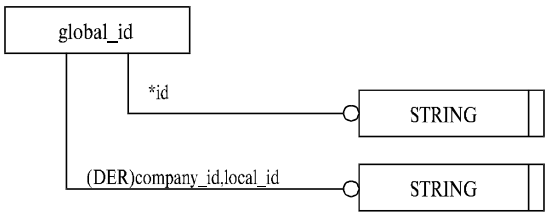


Figure 92 - ARM EXPRESS-G UoF Items diagram 2

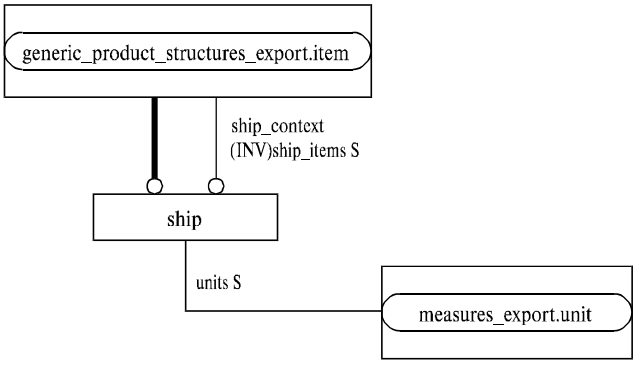


Figure 93 - ARM EXPRESS-G UoF Items diagram 3

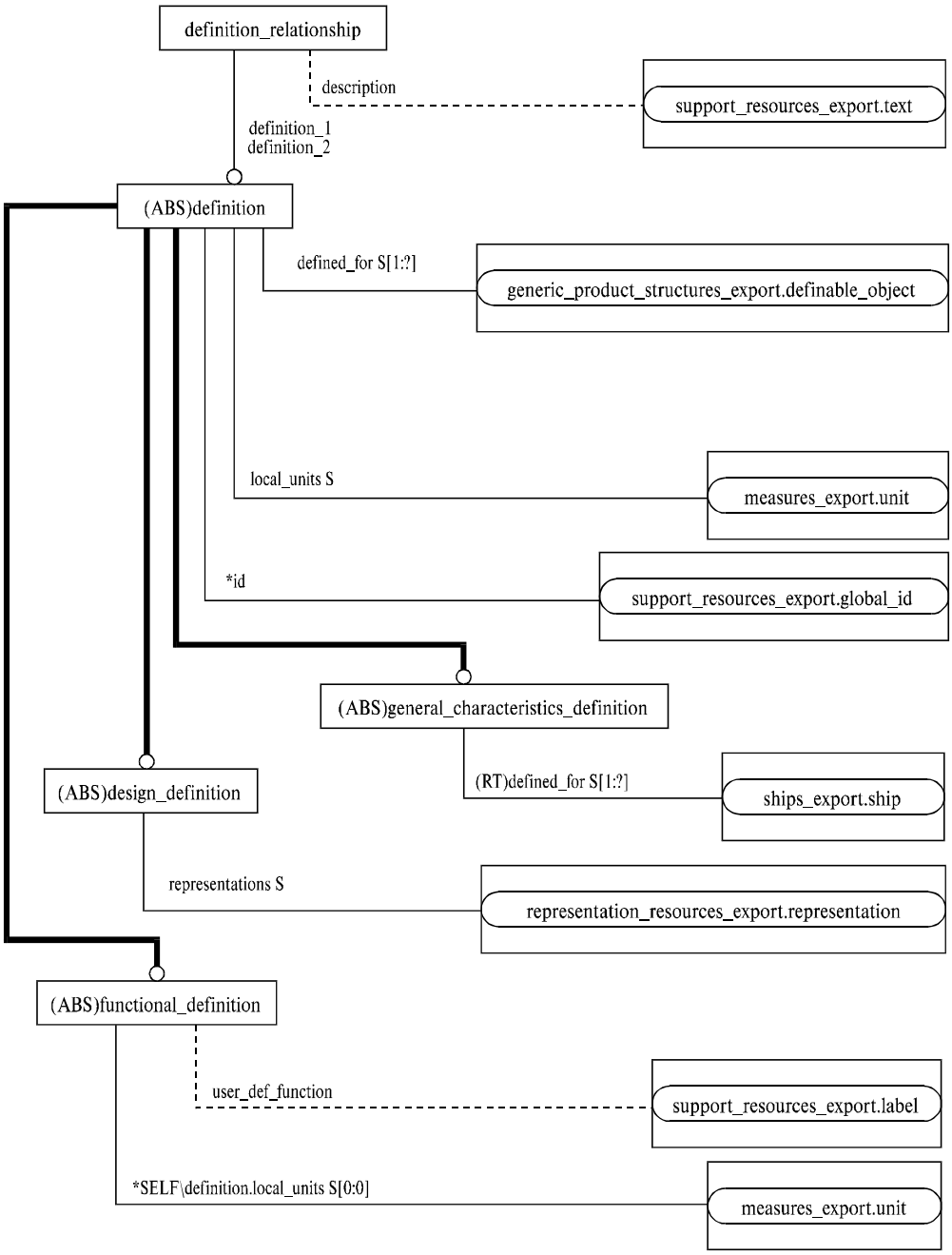


Figure 94 - ARM EXPRESS-G UoF Definitions

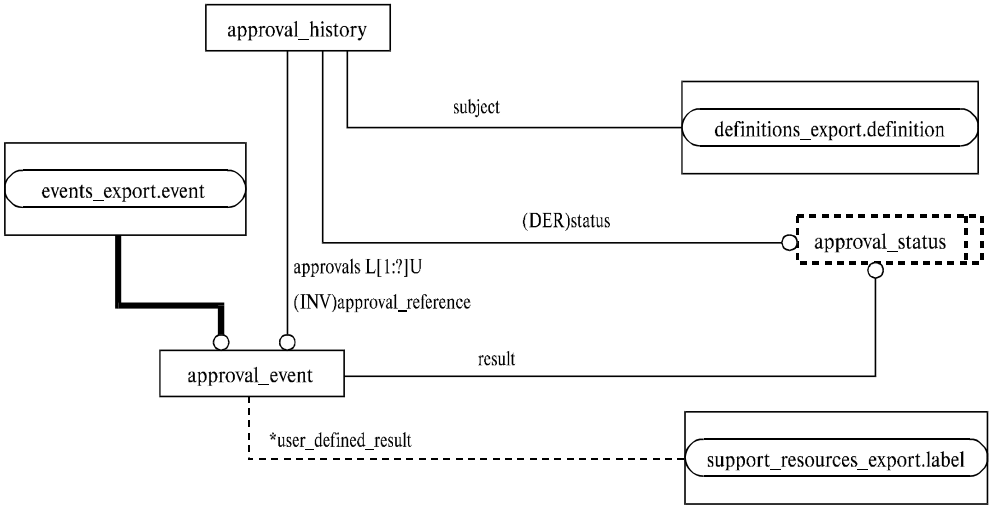


Figure 95 - ARM EXPRESS-G UoF Configuration management diagram 1

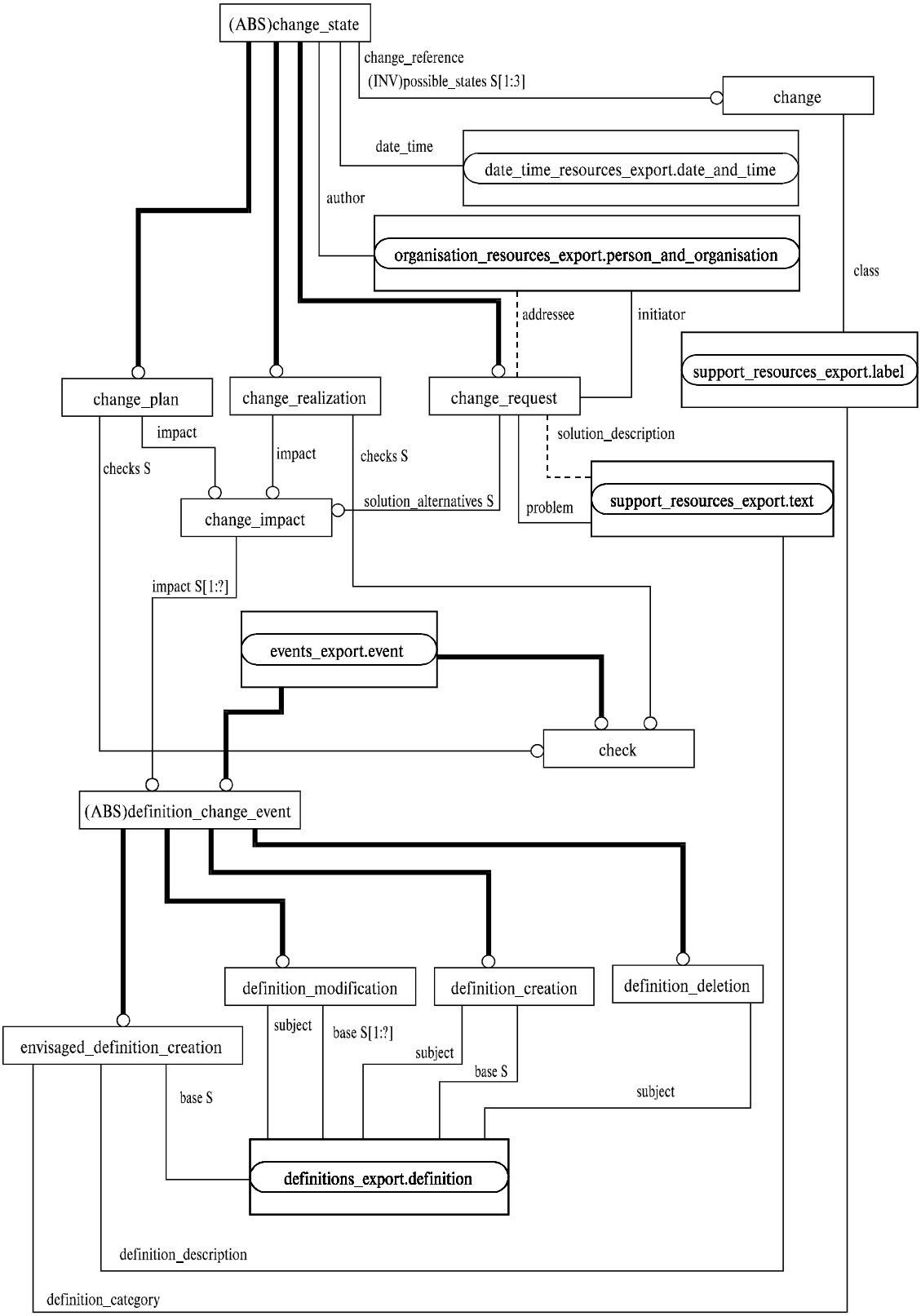


Figure 96 - ARM EXPRESS-G UoF Configuration management diagram 2

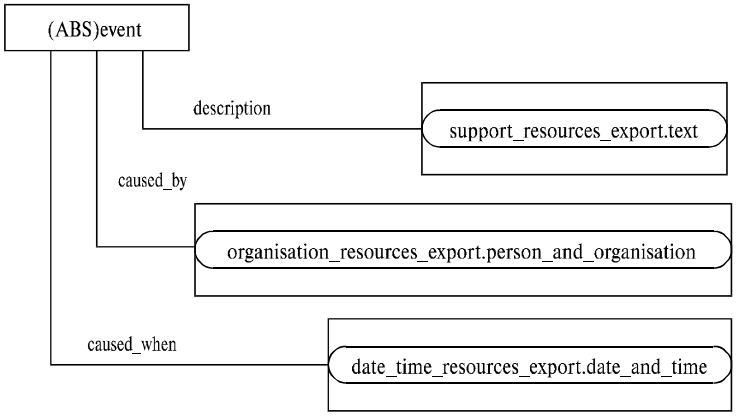


Figure 97 - ARM EXPRESS-G UoF Configuration management diagram 3

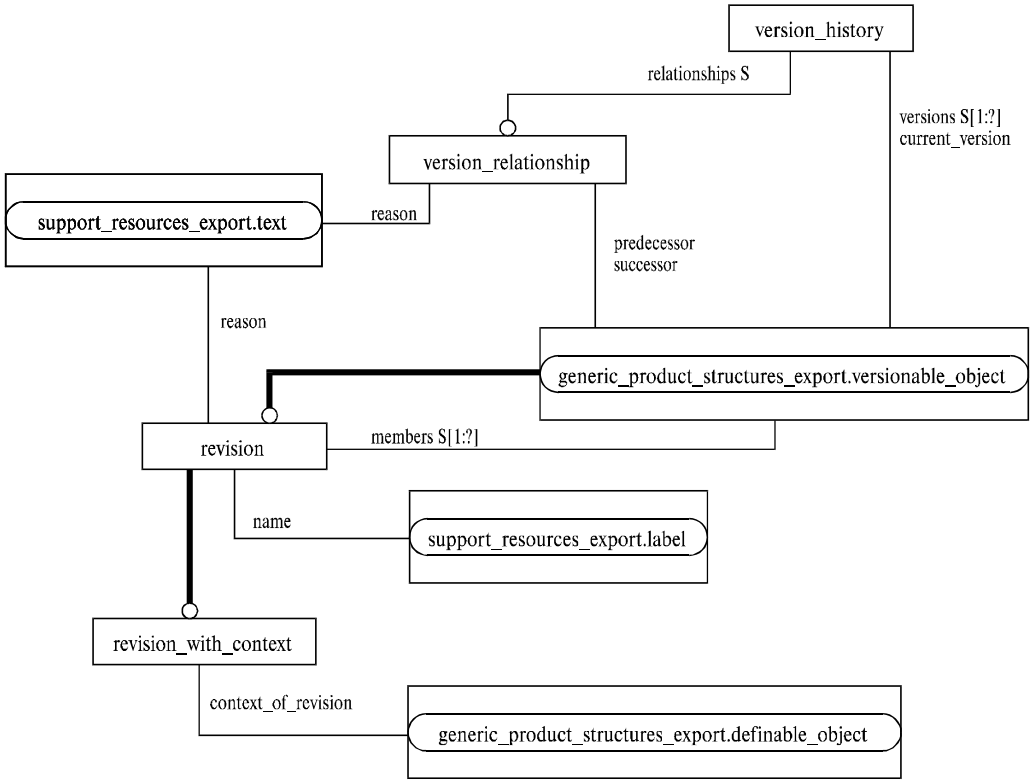


Figure 98 - ARM EXPRESS-G UoF Configuration management diagram 4

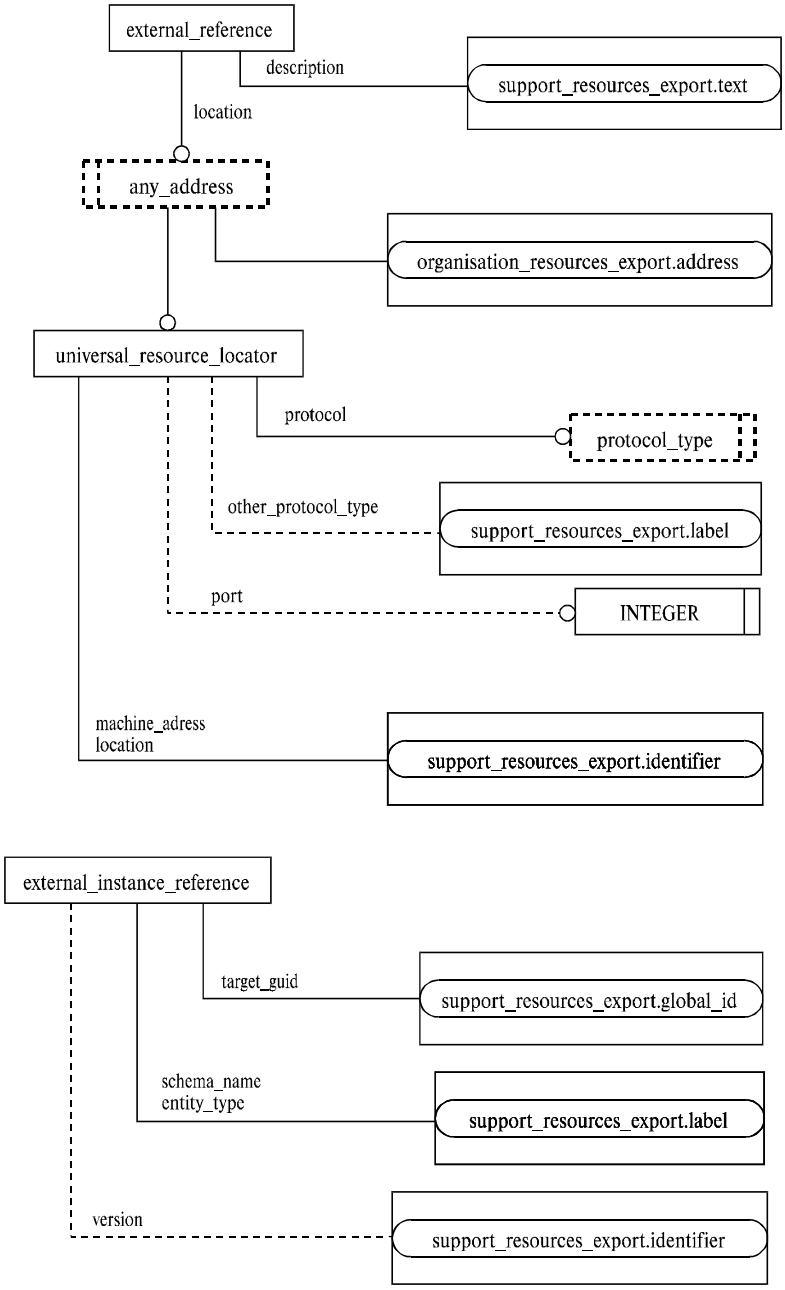


Figure 99 - ARM EXPRESS-G UoF External references diagram 1

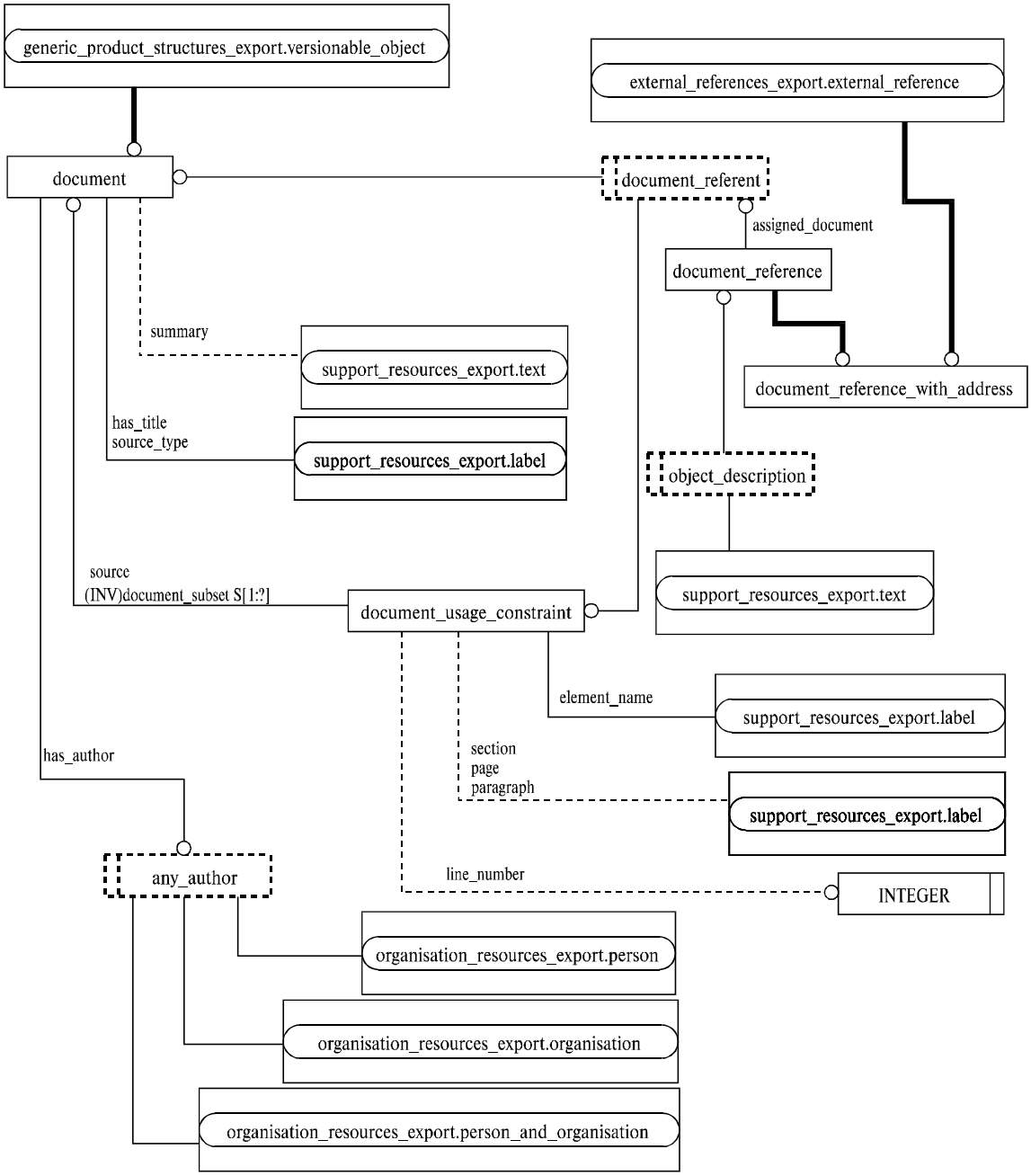


Figure 100 - ARM EXPRESS-G UoF External references diagram 2

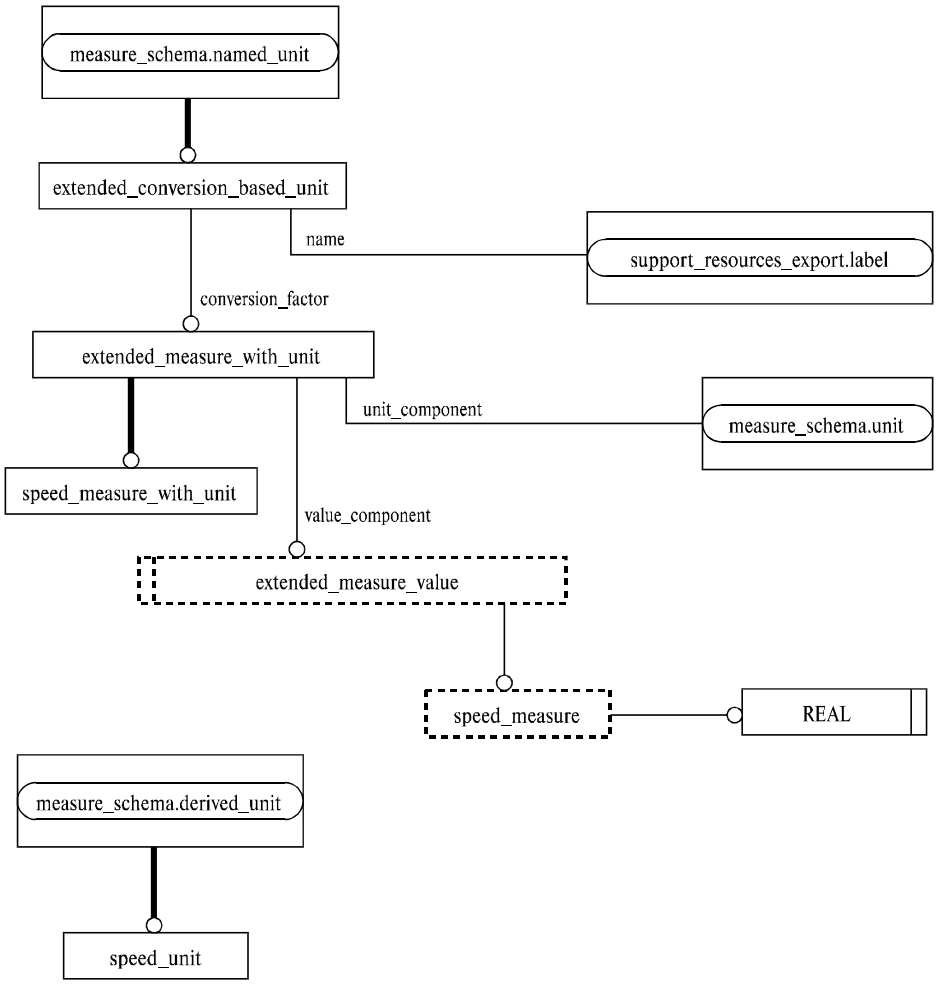


Figure 101 - ARM EXPRESS-G UoF Ship measures

Annex H
(informative)

AIM EXPRESS-G

The diagrams used in this annex correspond to the AIM EXPRESS expanded listing given in annex A. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

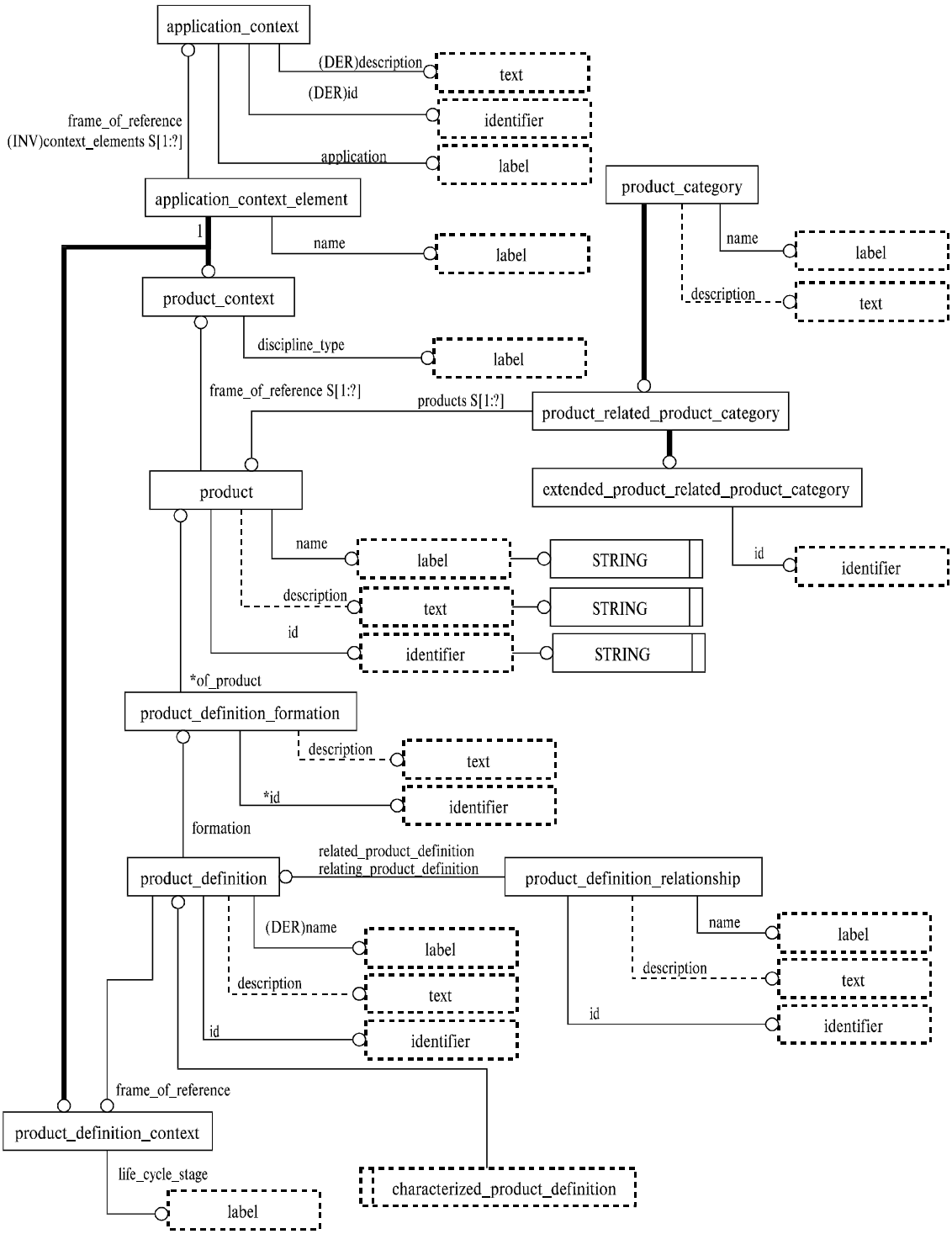


Figure 102 – AIM EXPRESS-G diagram 1 of 21

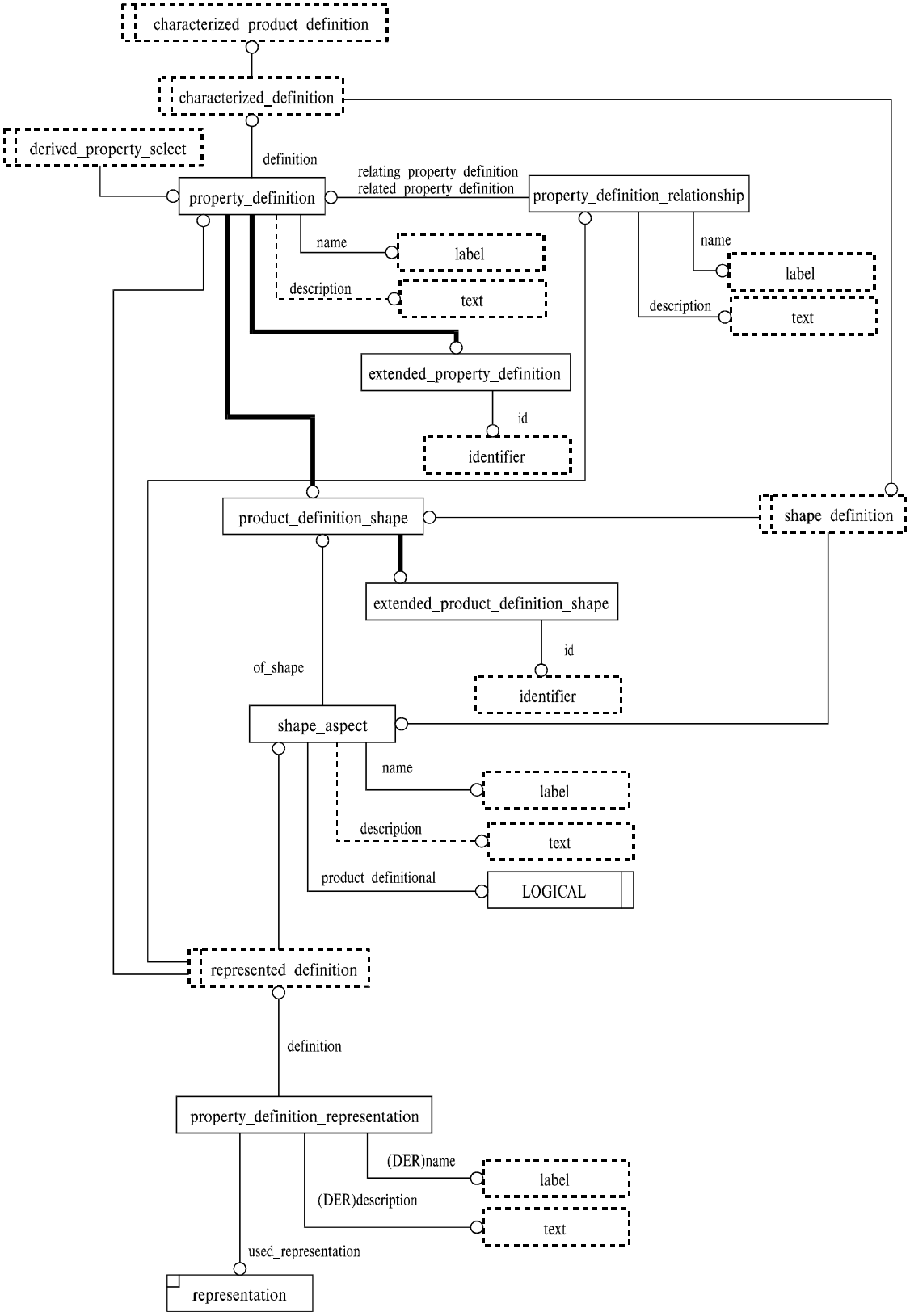


Figure 103 – AIM EXPRESS-G diagram 2 of 21

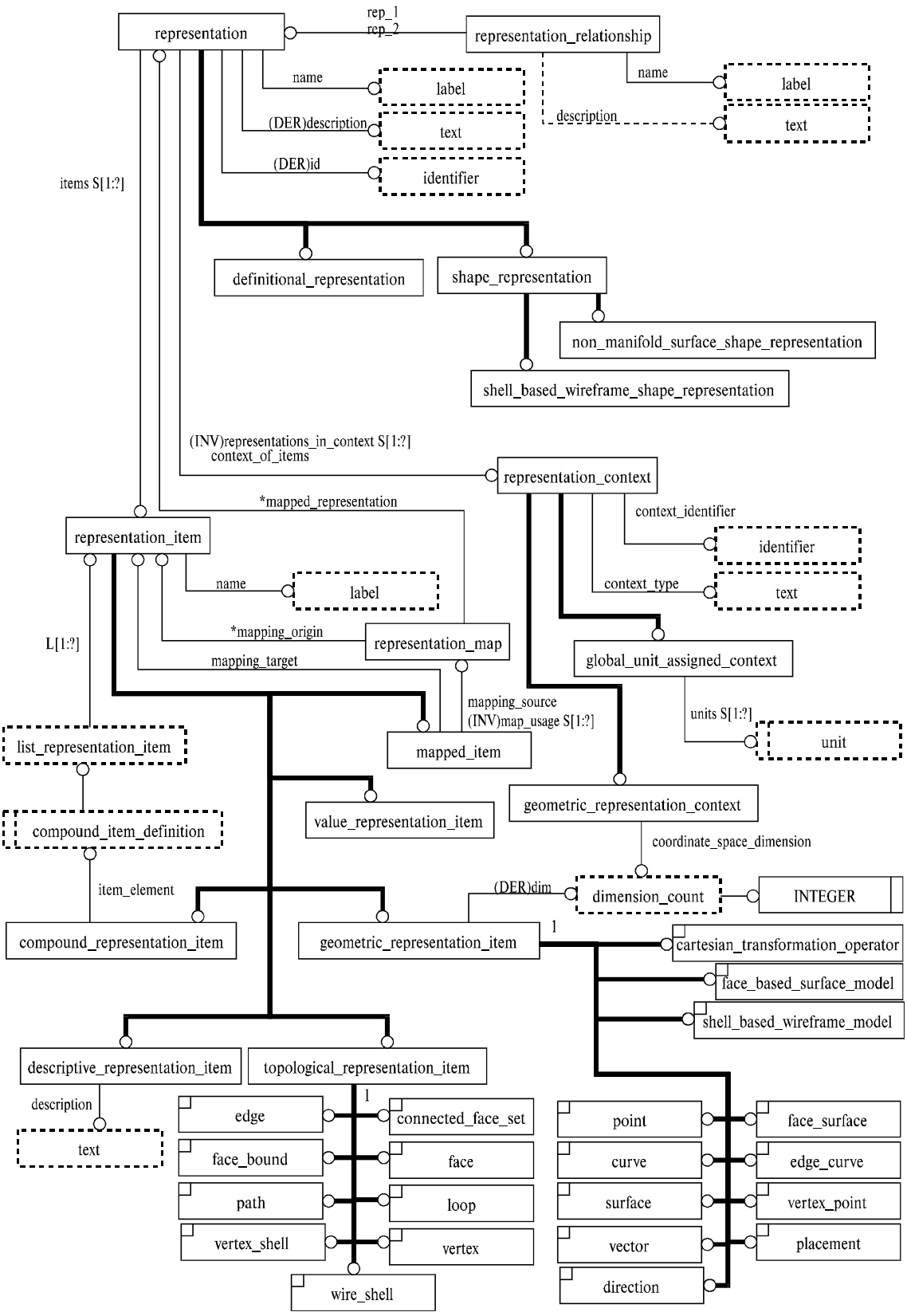


Figure 104 – AIM EXPRESS-G diagram 3 of 21

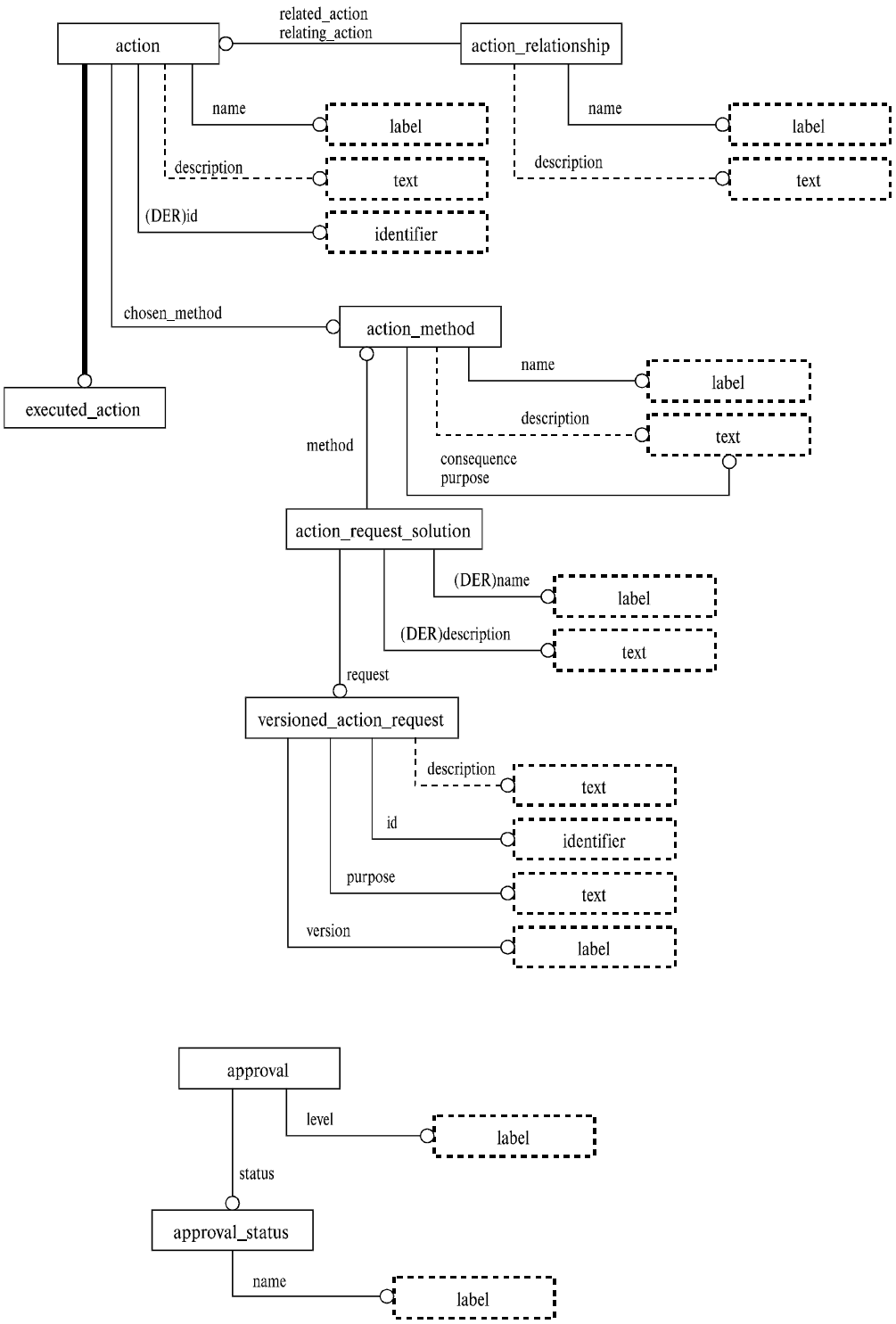


Figure 105 – AIM EXPRESS-G diagram 4 of 21

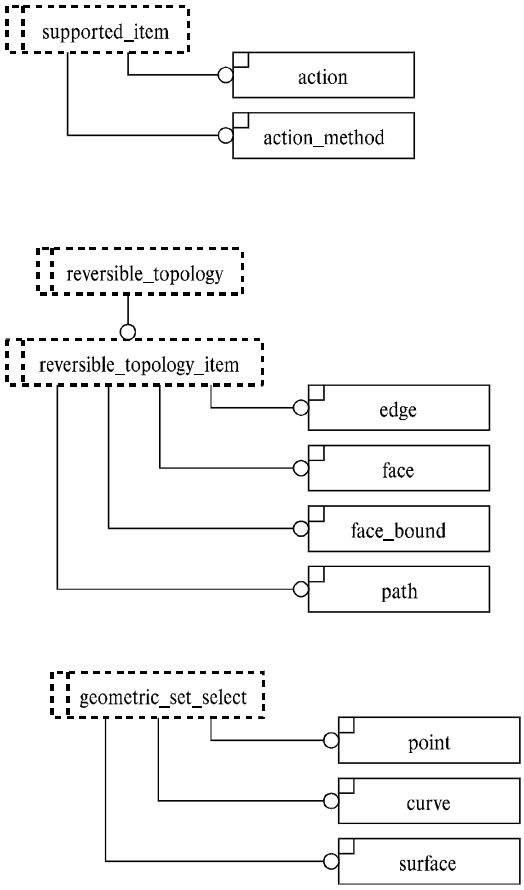


Figure 106 – AIM EXPRESS-G diagram 5 of 21

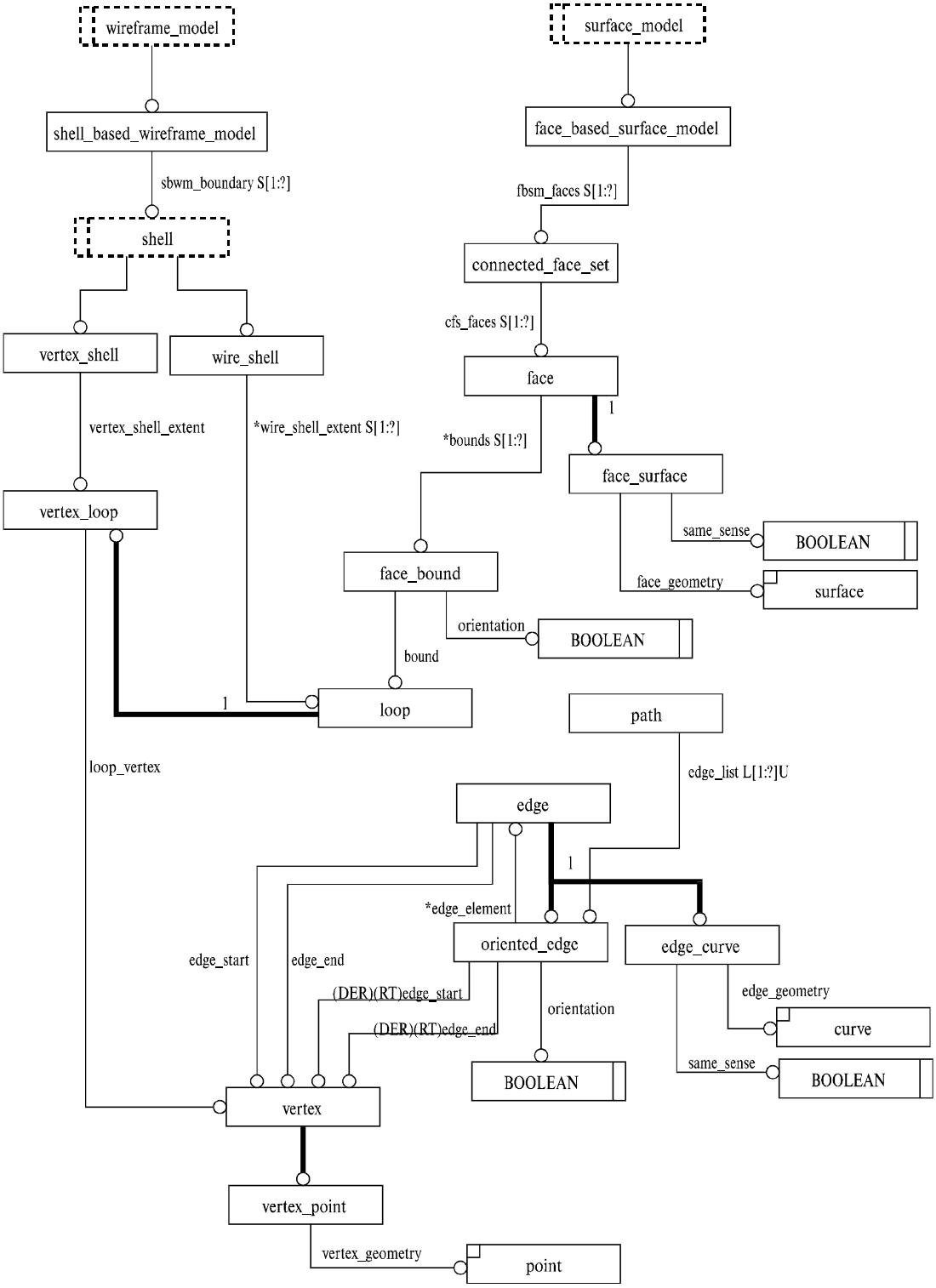
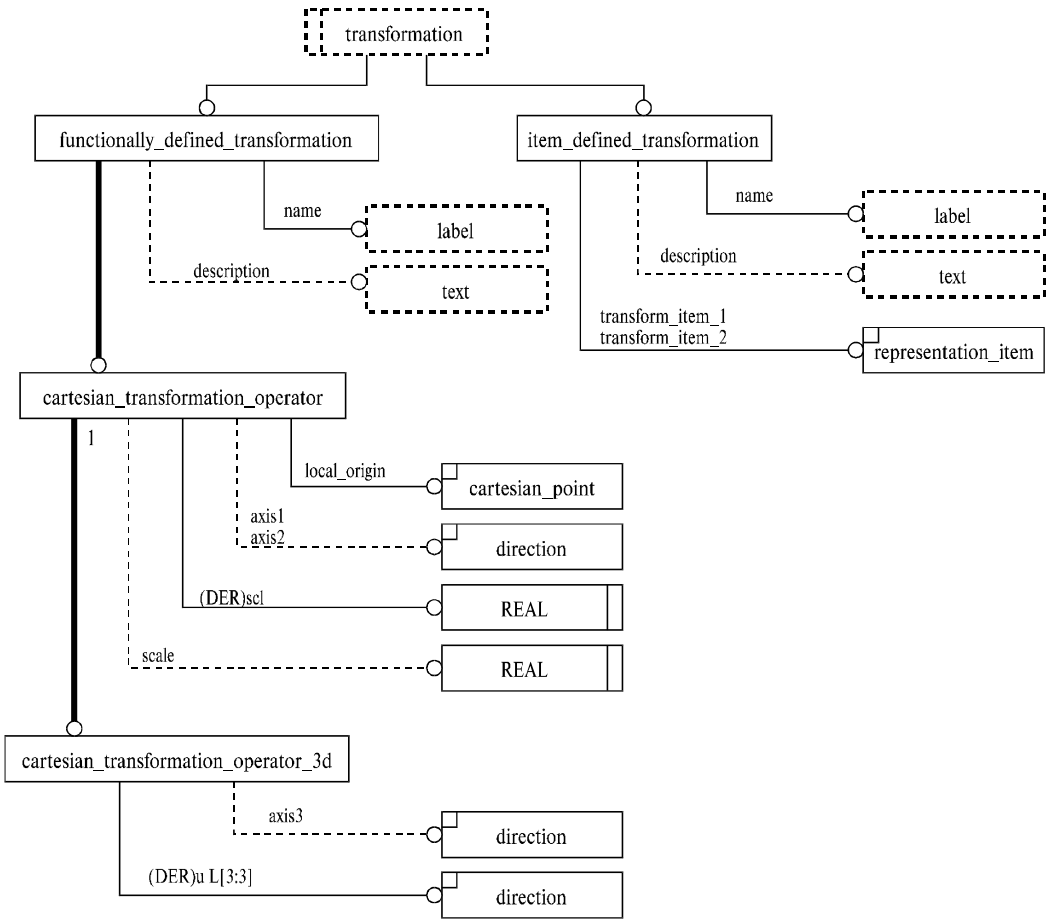


Figure 107 – AIM EXPRESS-G diagram 6 of 21



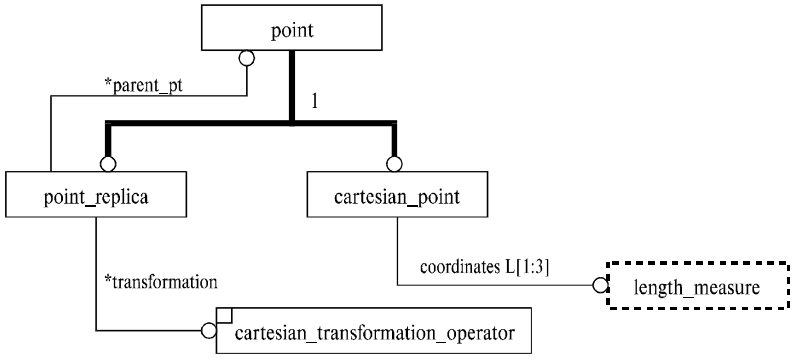


Figure 108 – AIM EXPRESS-G diagram 7 of 21

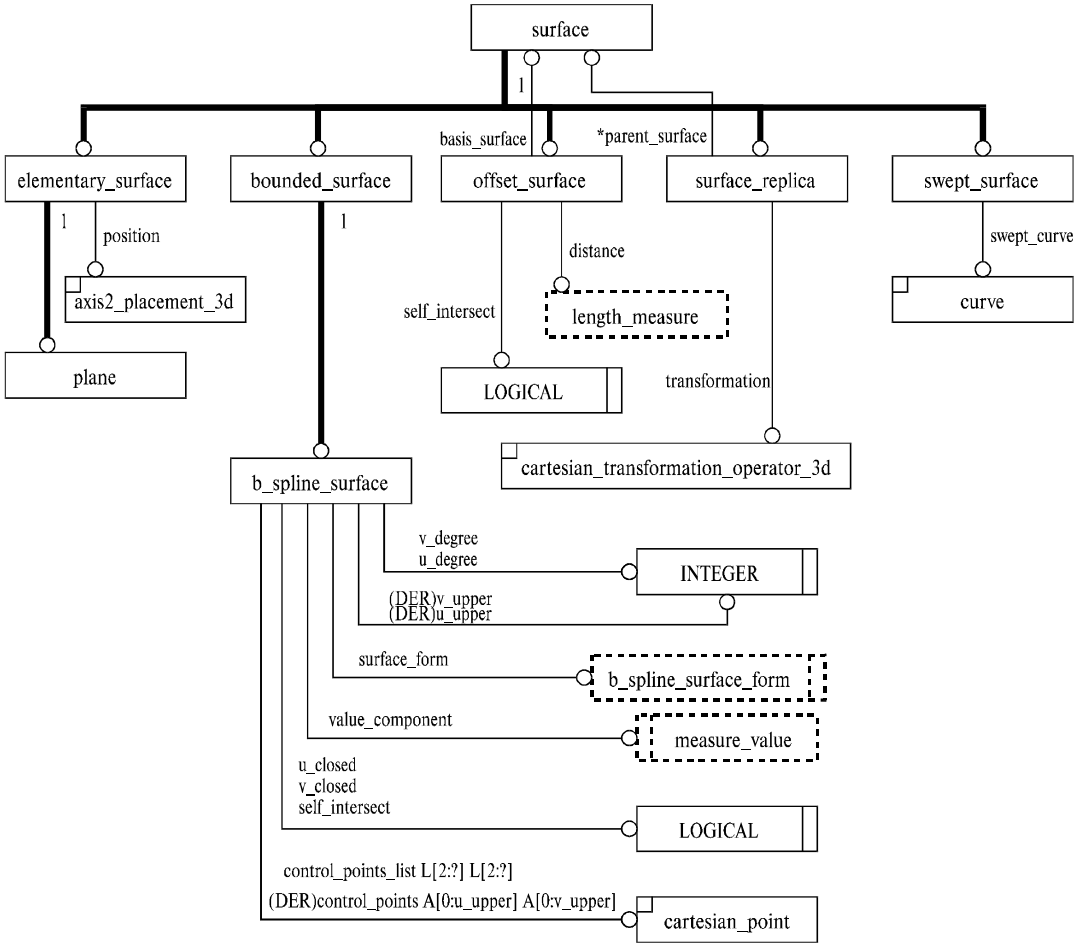


Figure 109 – AIM EXPRESS-G diagram 8 of 21

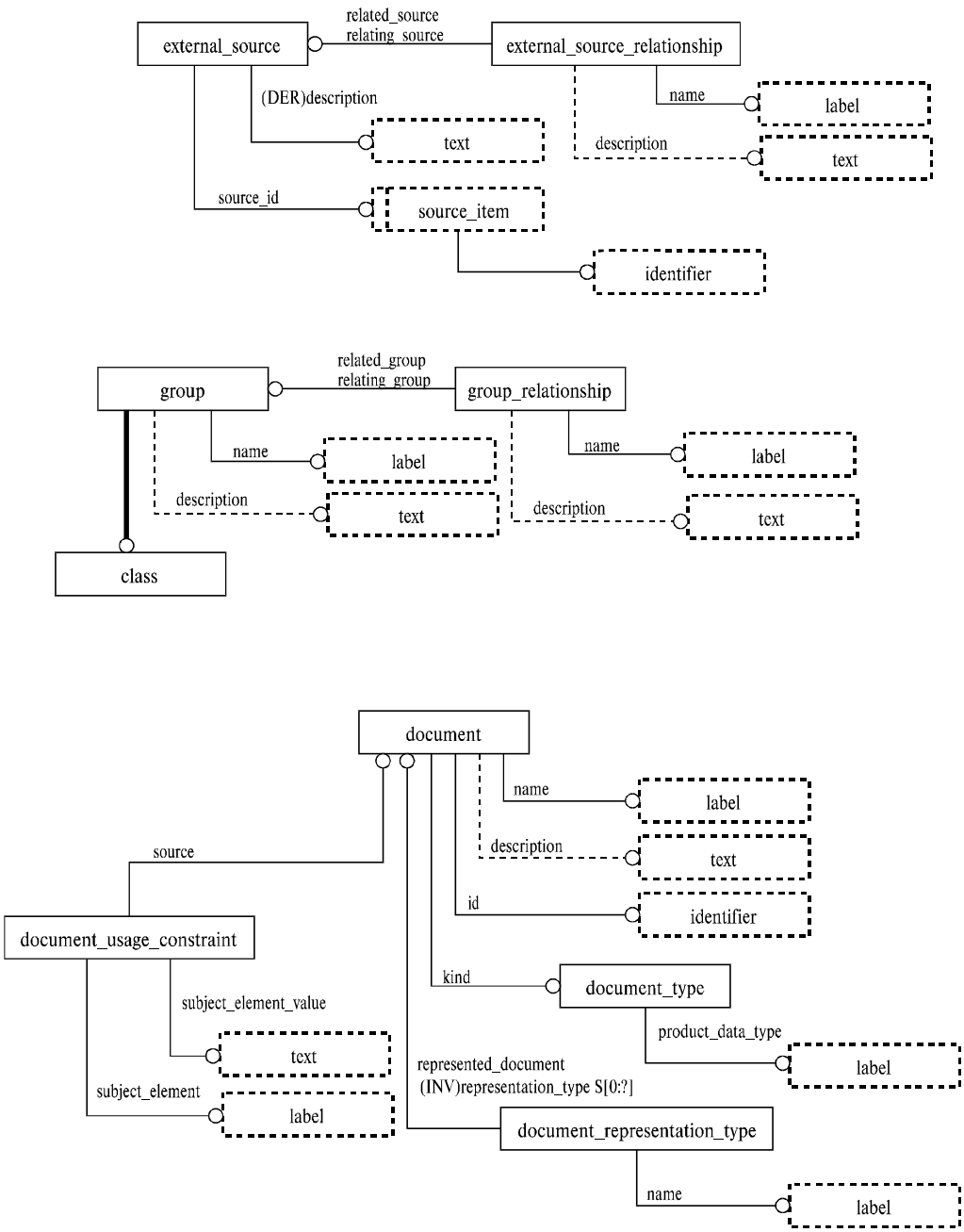


Figure 110 – AIM EXPRESS-G diagram 9 of 21

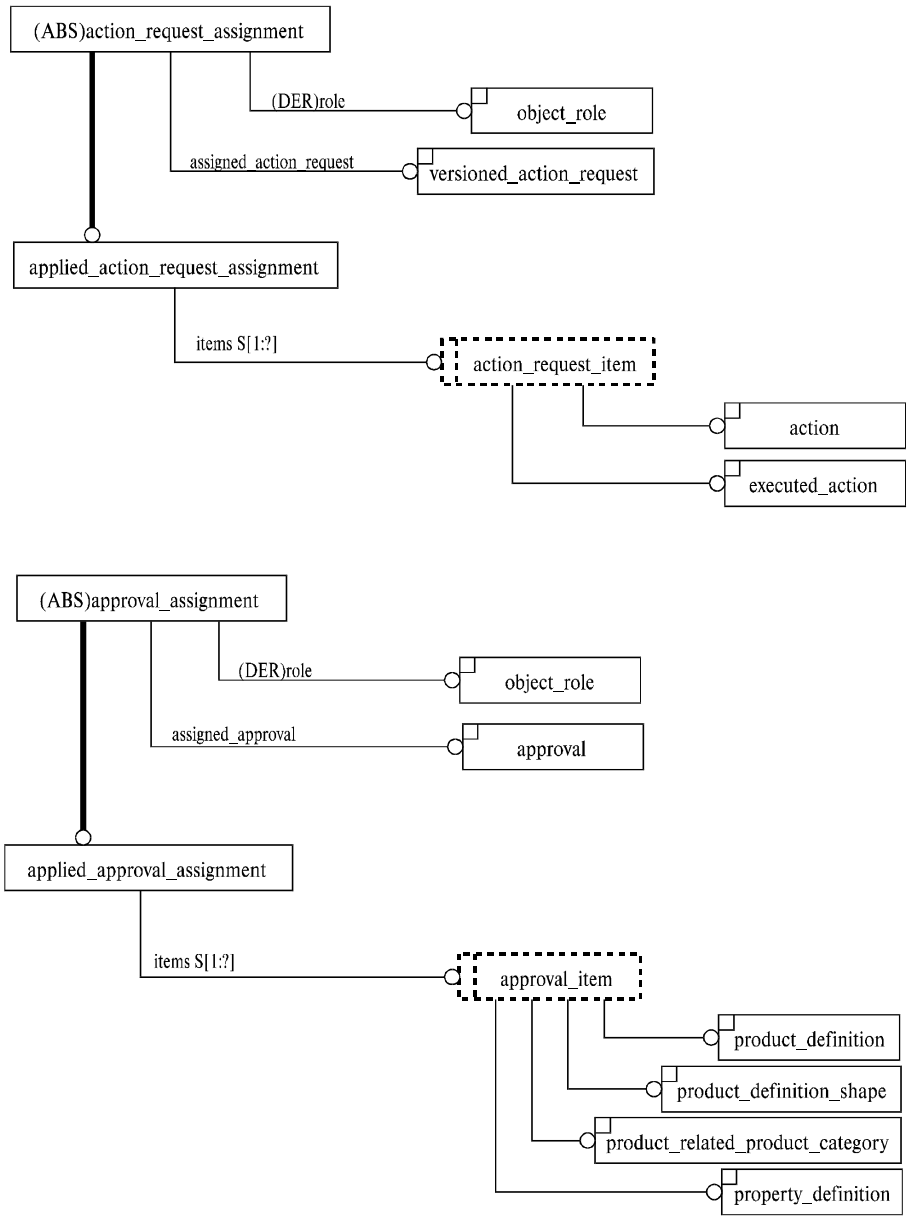


Figure 111 – AIM EXPRESS-G diagram 10 of 21

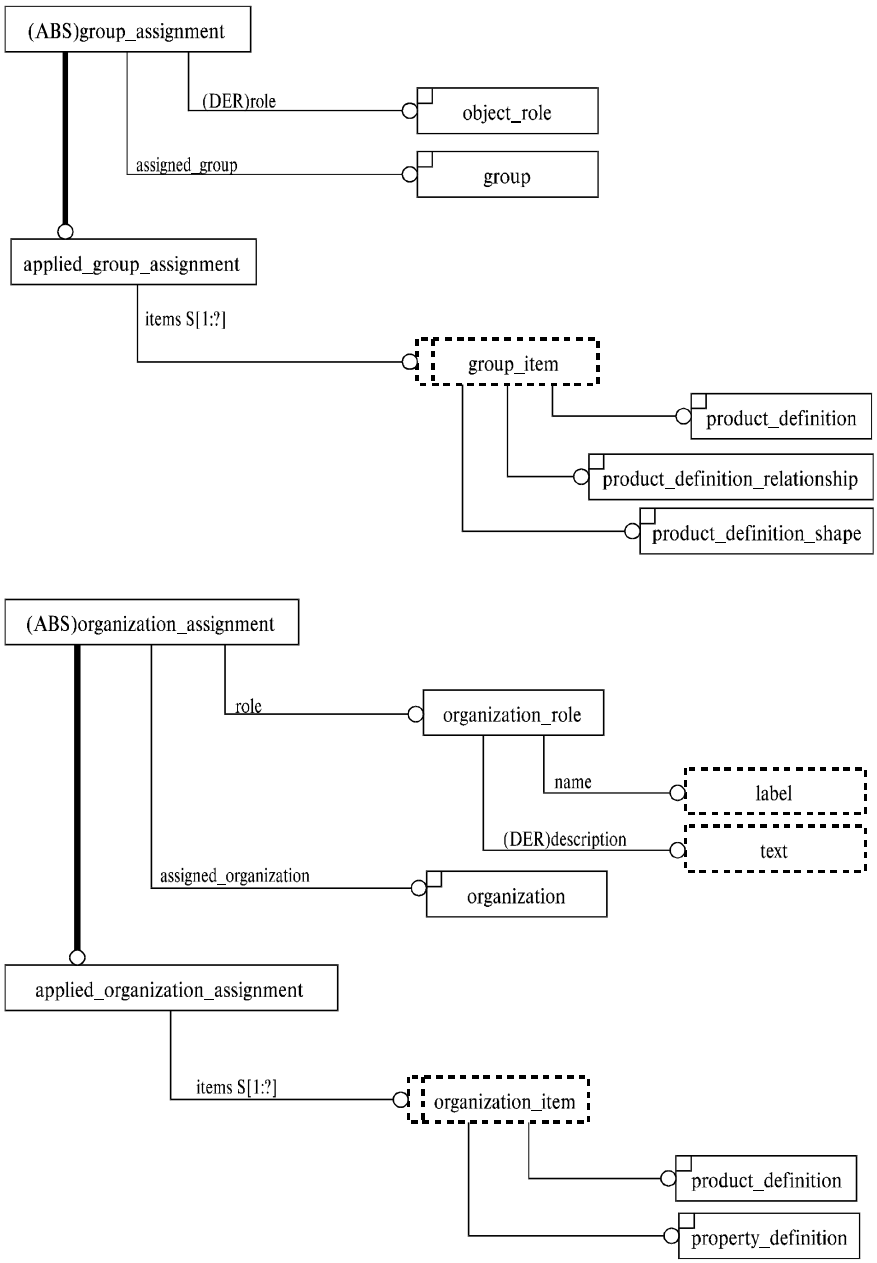


Figure 112 – AIM EXPRESS-G diagram 11 of 21

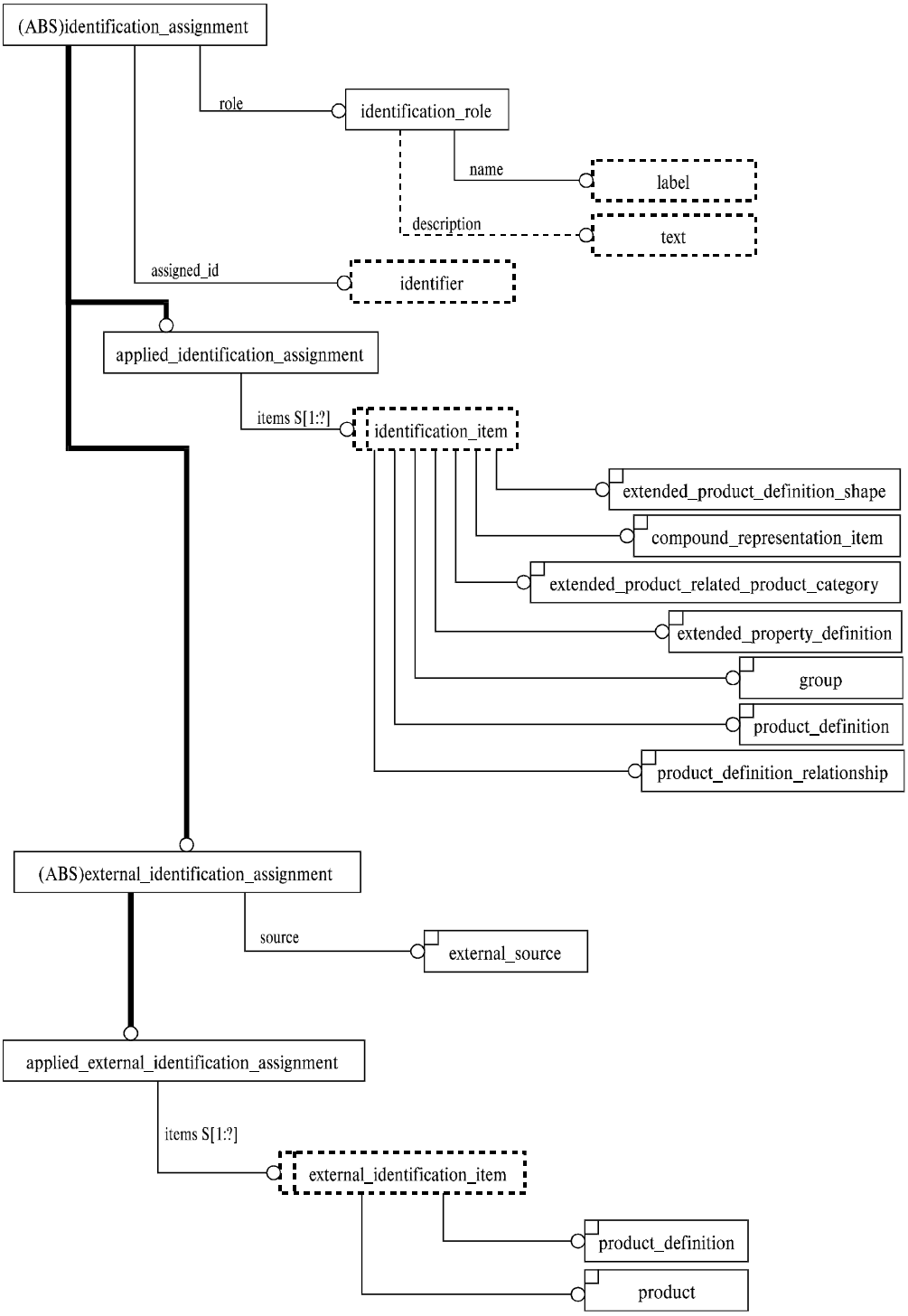


Figure 113 – AIM EXPRESS-G diagram 12 of 21

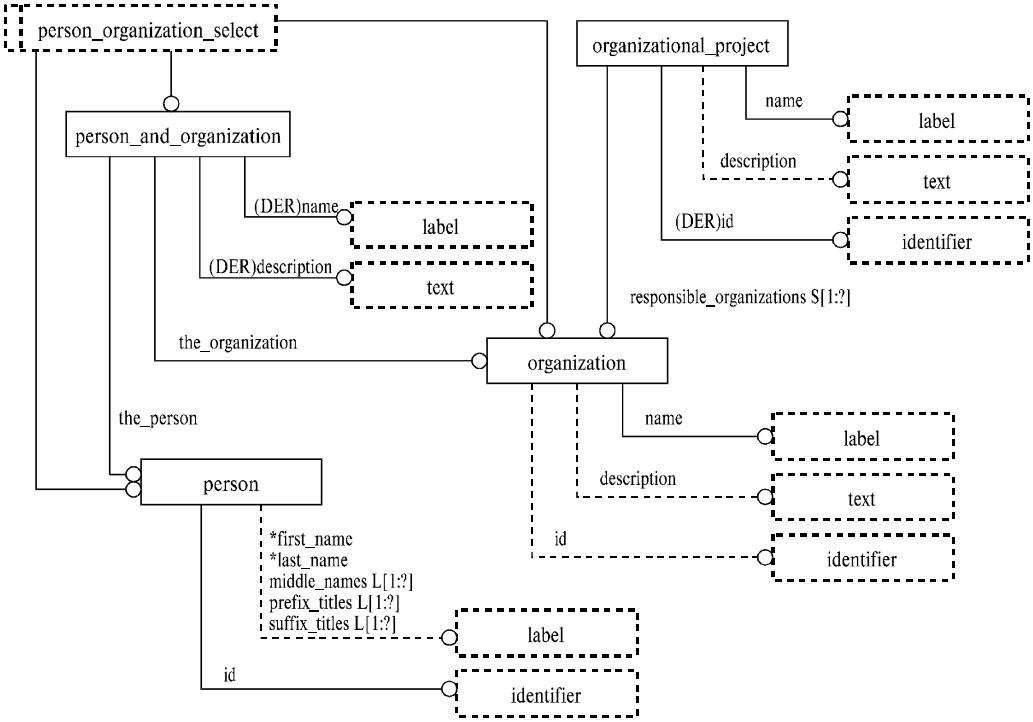


Figure 114 – AIM EXPRESS-G diagram 13 of 21

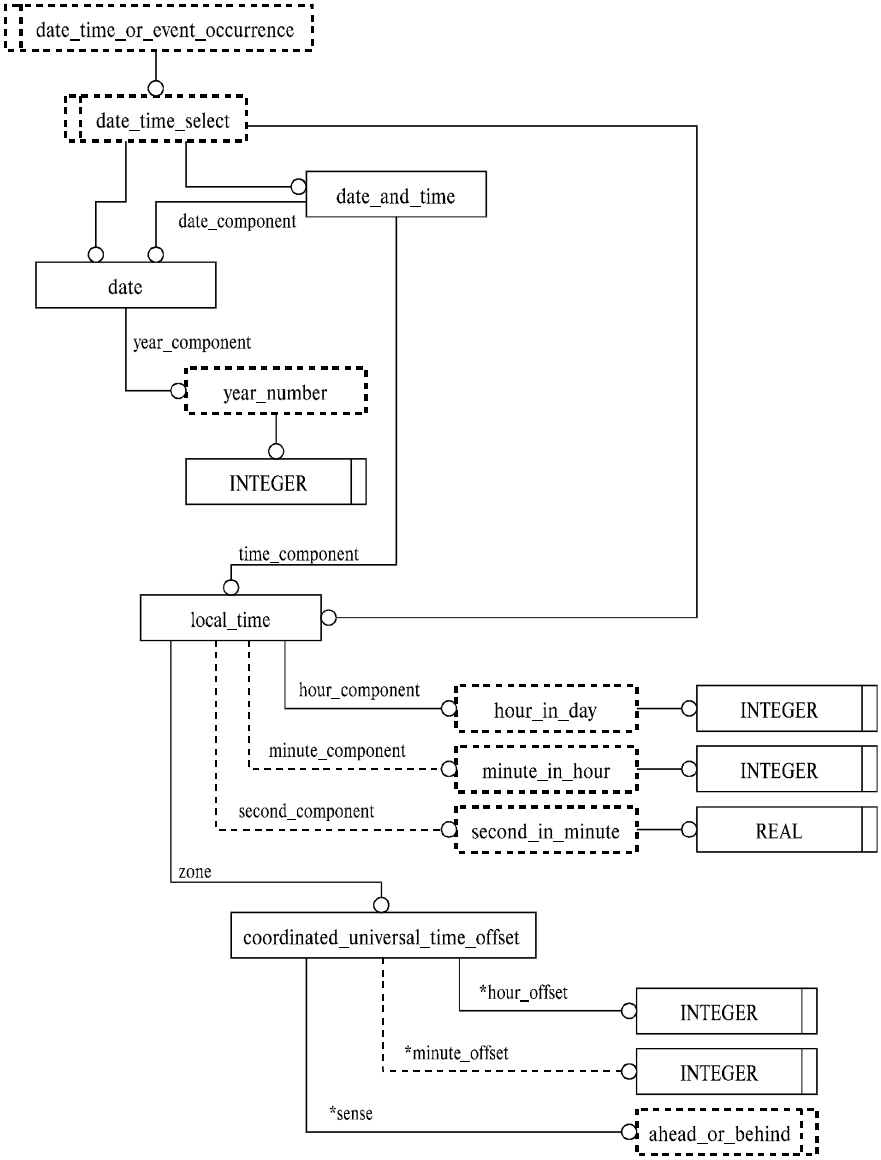


Figure 115 – AIM EXPRESS-G diagram 14 of 21

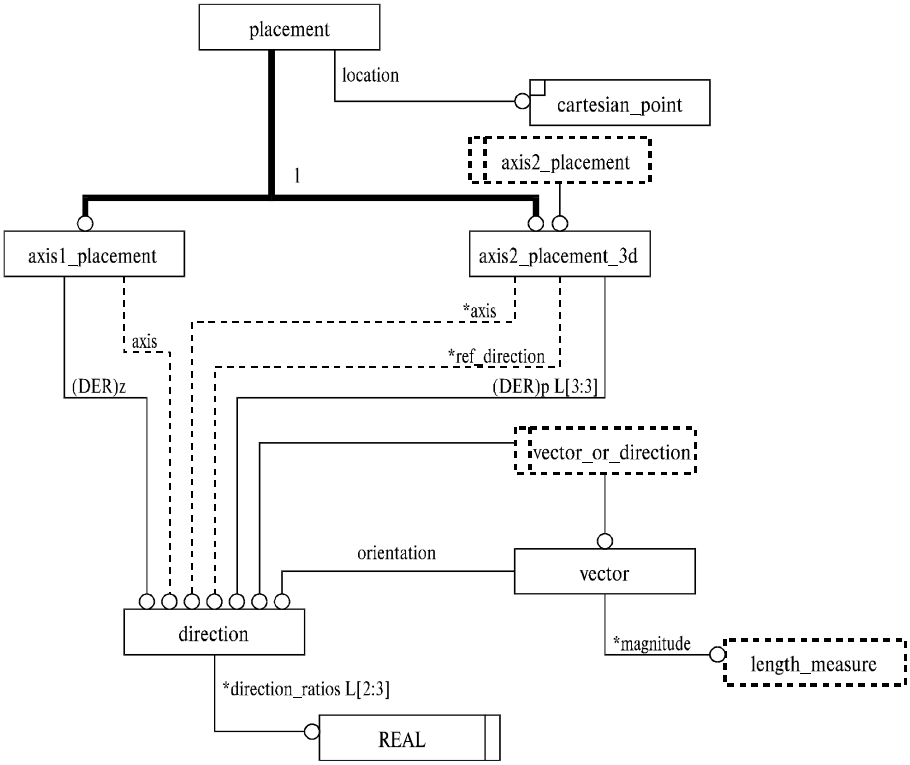


Figure 116 – AIM EXPRESS-G diagram 15 of 21

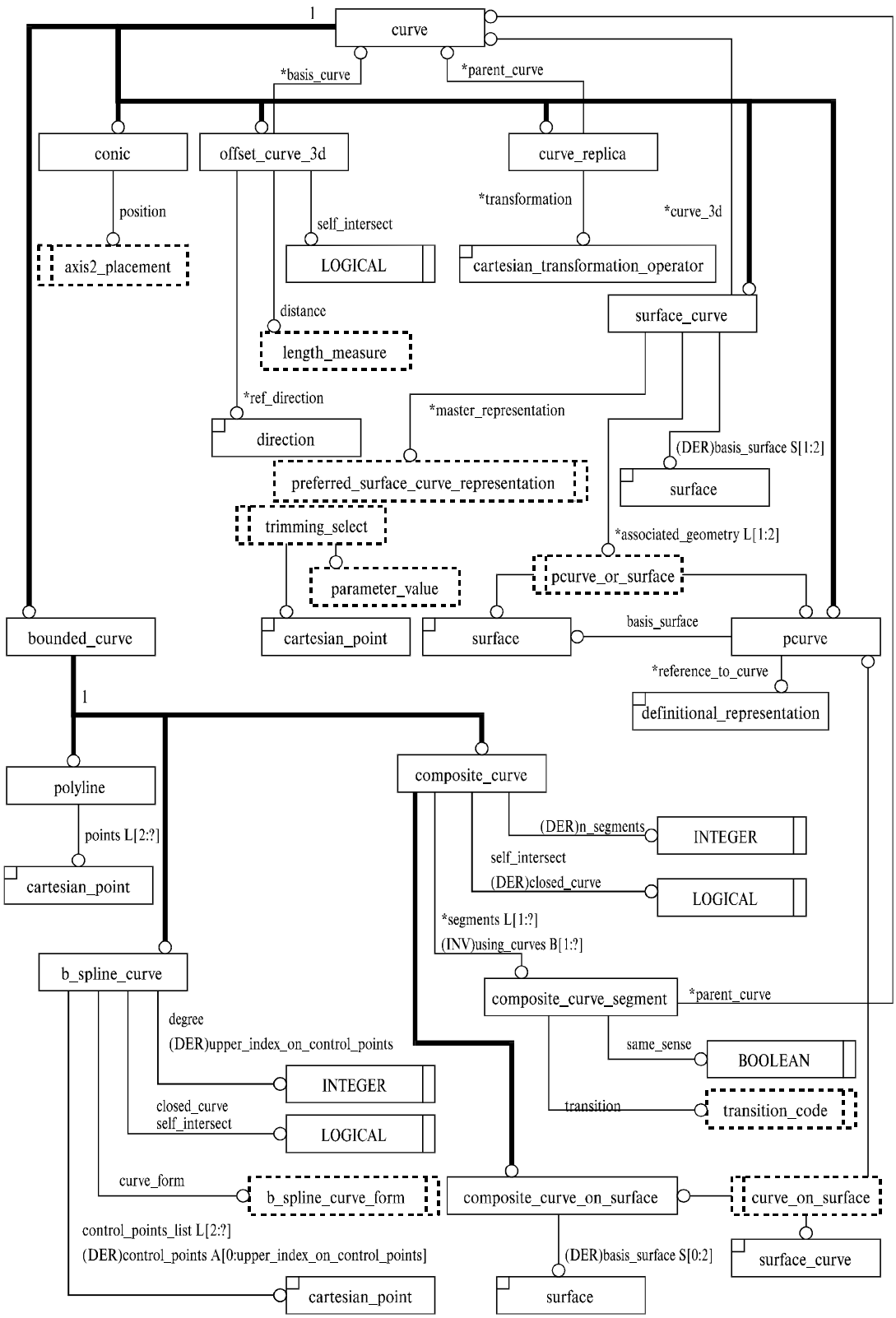


Figure 117 – AIM EXPRESS-G diagram 16 of 21

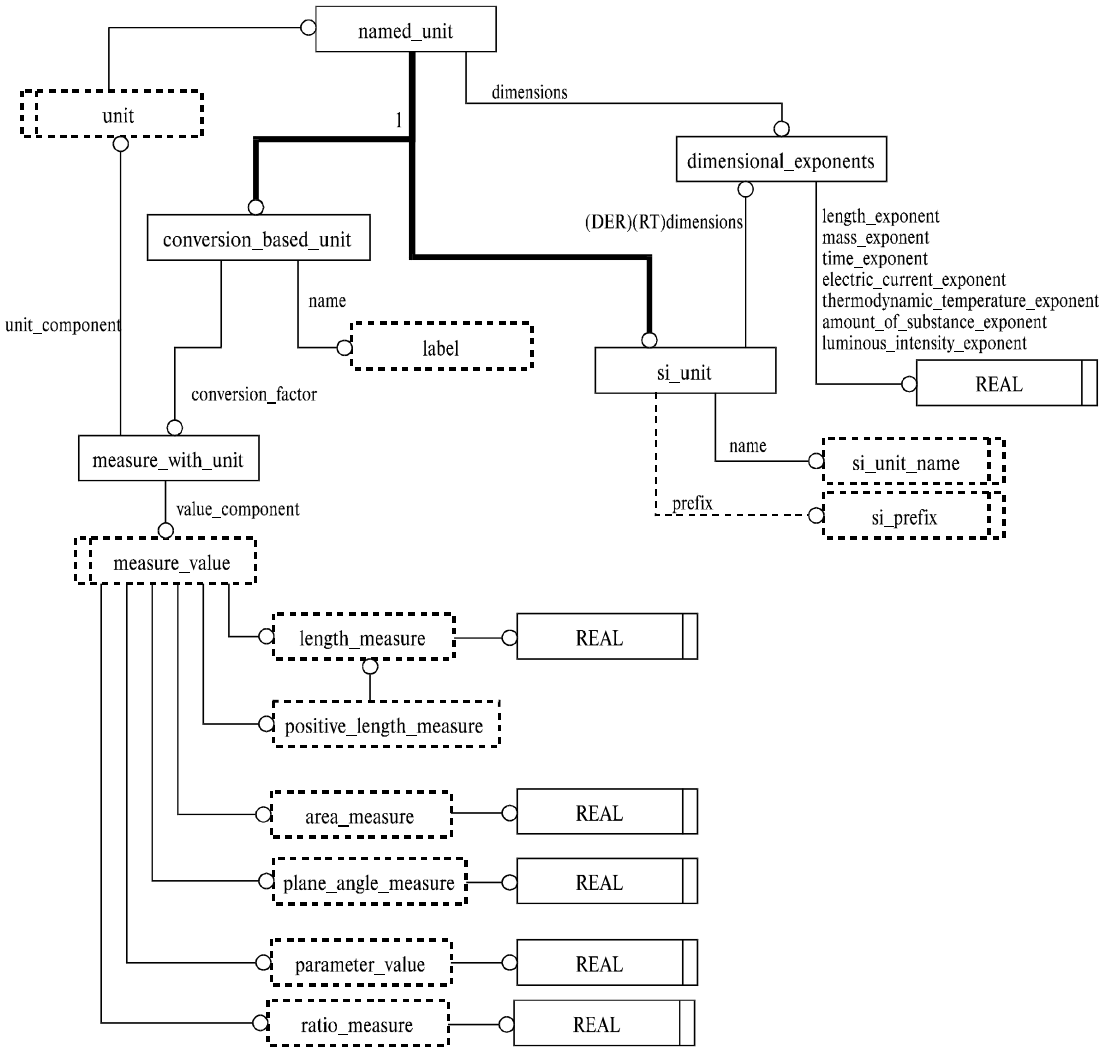


Figure 118 – AIM EXPRESS-G diagram 17 of 21

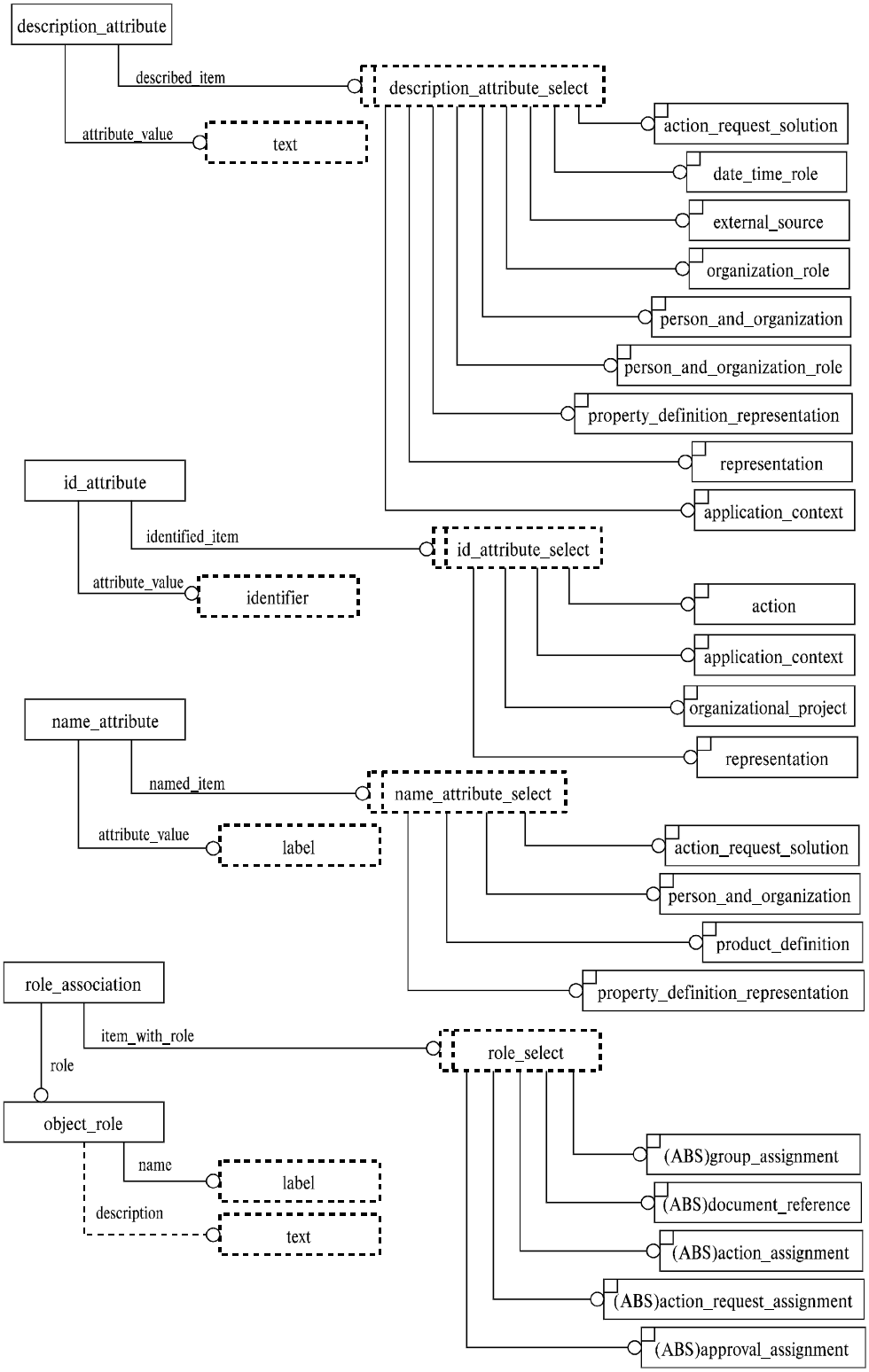


Figure 119 – AIM EXPRESS-G diagram 18 of 21

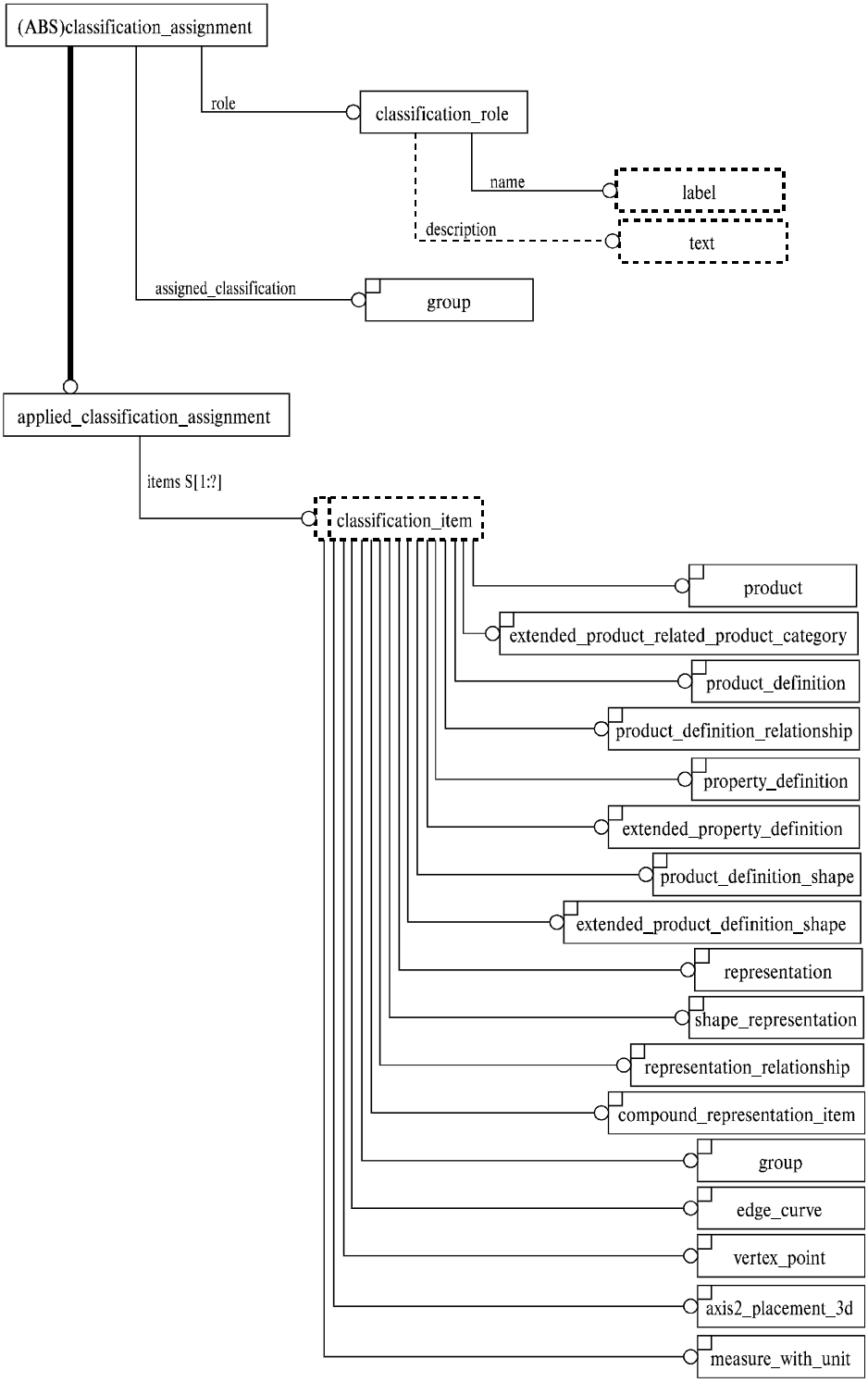


Figure 120 – AIM EXPRESS-G diagram 19 of 21

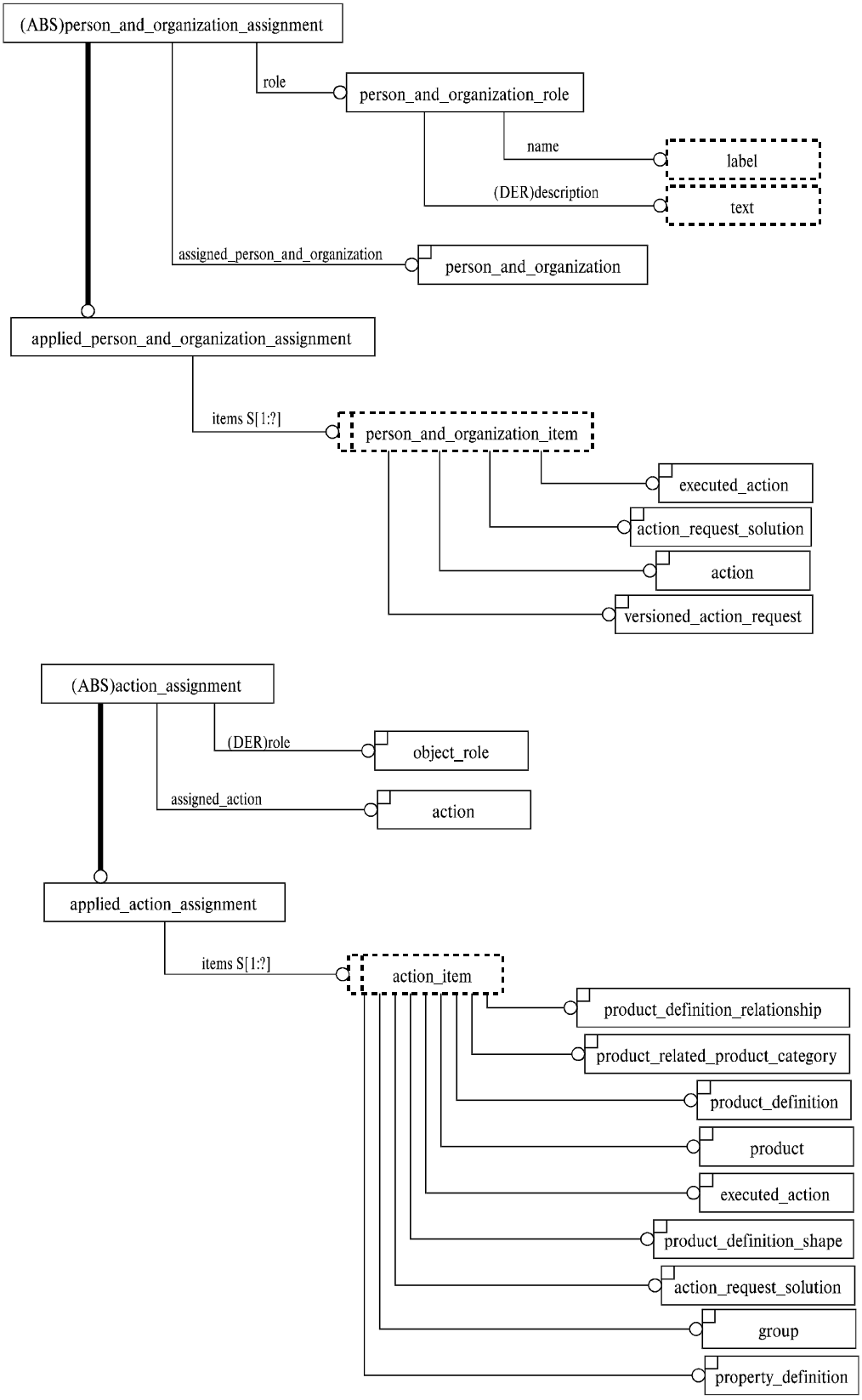


Figure 121 – AIM EXPRESS-G diagram 20 of 21

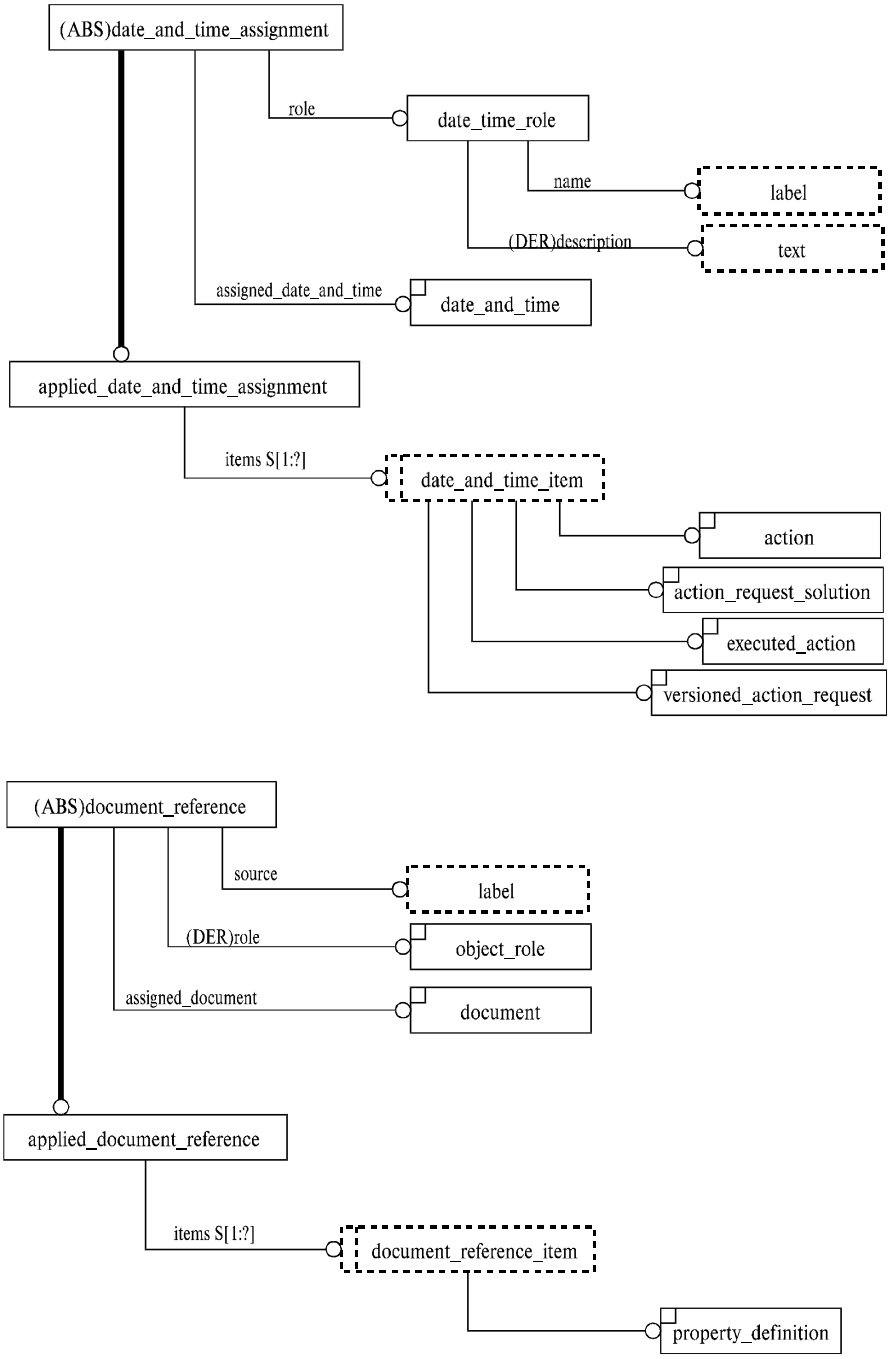


Figure 122 – AIM EXPRESS-G diagram 21 of 21

Annex J
(informative)

Computer interpretable listings

It also provides a listing of each EXPRESS schema specified in this part of ISO 10303 without comments or other explanatory text. These listings are available in computer-interpretable form and can be found at the following URLs:

Short names: http://www.mel.nist.gov/div826/subject/apde/snr/

EXPRESS: http://www.mel.nist.gov/step/parts/part216/cd/

If there is difficulty accessing these sites contact ISO Cenral Secretariat or contact the ISO TC 184/SC4 Secretariat directly at: sc4sec@cme.nist.gov.

NOTE - The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex K
(informative)

AP 216 Ship Moulded Forms usage guide

This chapter will answer some questions to the ARM of this part of ISO 10303.

K.1 How can a spacing position be related to a moulded form?

Use a local_co_ordinate_system_with_position_reference pointing to this moulded_form.

K.2 How can an externally defined moulded form be referenced?

The list of moulded forms in the ship moulded form includes an externally defined mouled form, identified by the external reference and in particular by the global id.

K.3 What happens with the co-ordinate systems when the ship is in water at a specific floating position

The ship has a fixed global co-ordinate system. For hydrostatic calculations a local co-ordinate system is defined which is fixed to the water surface. The parent co-ordinate system is the global co-ordinate system.

K.4 How will can parts of a ship hull be modelled?

EXAMPLE - a fore and aft hull.

Two instances of moulded form with a functional definition ship hull and a user defined function fore hull and aft hull will be generated. Both moulded forms are part of a ship moulded form and linked via a moulded form relationship.

ship moulded form

moulded form relationship

2 moulded forms

Annex L
(informative)

Technical discussions

L.1 The ship product model

The aim of STEP is the support of methods and functionality to develop product models. A product model is the unambiguous representation of a product, such as a ship, automobile, aeroplane or production plant, in a computer interpretable neutral format, throughout the lifetime of the product. The goal is to represent a product from the requirement definition and conceptual design stages through to production, maintenance and finally decommissioning and dismantling. The representation is intended to include all geometric and non-geometric data associated with the product including topology, geometry, functionality, strength, parameter and characteristics, condition status and service history.

STEP is based on the idea that the master representation of a product is related to its product structure. That means, to define a ship within STEP, basic identification information and ship structure information has to be provided. As part of this approach, geometry is taken as a particular type of representation of the ship and its structure. The consequence of this is that when the moulded form geometry is exchanged using STEP additional information on how each element represented by the geometry relates to a particular part of a particular ship. This is different to existing and previous exchange standards where only the pure geometry is exchanged. This is the reason why STEP is product based and not drawing or geometry based. It is also the reason for using STEP as a set of standard descriptions for reusable product based data, the total collection of which are referred to as the product data model. STEP has been developed in conjunction with its own methods, so that it can be implemented in file exchange or databases alike.

In view of the complexity of the ship it is unavoidable to subdivide the STEP product model for the ship into recognisable and usable parts. In consequence of this in 1993 an Application Protocol planning model was agreed. This model divides the shipbuilding domain into seven distinct functional areas, allowing for partition into a fixed number of Application Protocols. Since then the model has been slightly updated. The current version of the ship product model is shown schematically in Figure 123. The key elements of the ship product model are:

— Arrangements,

— Moulded forms,

— Mechanical systems (machinery, propulsion, cargo handling),

— Structures,

— Distribution systems (piping, HVAC, electrical, hydraulics/pneumatics),

— Outfitting & furnishings,

— Communication,

— Combat systems,

— Navigation,

— Operation.

Each functional area of the ship product model is described by one or more different application protocols. The shipbuilding application protocols:

— AP 215 Ship Arrangements,

— AP 216 Ship Moulded Forms,

— AP 217 Ship Piping Systems,

— AP 218 Ship Structures,

— AP 226 Ship Mechanical Systems,

which are currently under development at ISO level are marked with a grey background in Figure 123. The subdivision of the entire ship product model allows a distributed modelling work. It is also possible then to start the modelling work with the functional areas reflecting the early stages of the life cycle of the ship like ship moulded forms and validate these models before starting the modelling work at areas in the later lifecycle stages.

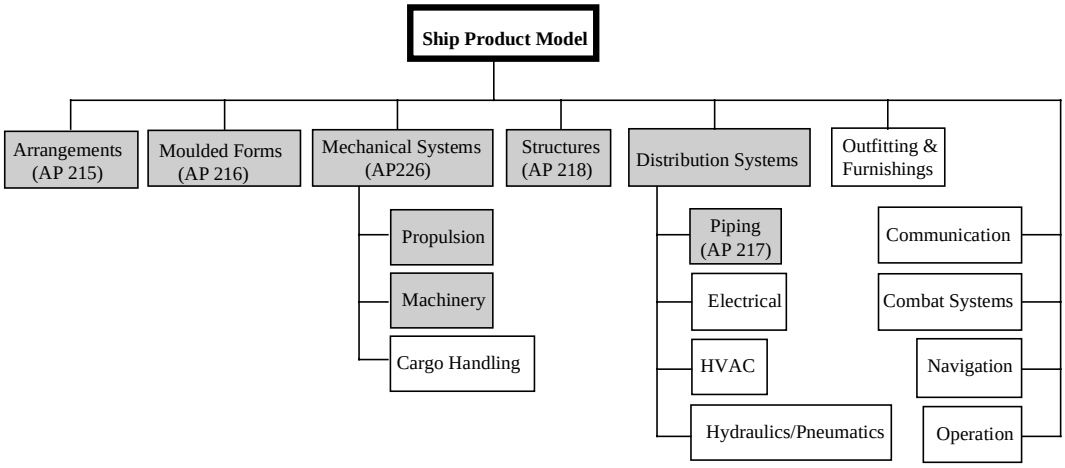


Figure 123 - Ship Product Model

Application protocols are the only parts of STEP that are intended to be used for a data exchange. But nevertheless each shipbuilding AP covers only a part of the ship product model. The consequence of this is that if the AP's should work together as an entire product model for the ship, then an overall mechanism, around which all the shipbuilding AP's can be integrated, has to be defined. Only then the product model can be implemented in a product data management system PDM- system, where all data are hold outside of application systems in the neutral data format of the ship product model. This mechanism for integrating different AP's is called the Ship Common Model SCM, which is described in the following chapter.

L.2 The Ship Common Model

The Ship Common Model was developed during the MARITIME project and defines a common framework and modelling basis for all shipbuilding AP's to ensure interoperability between these AP's. Unfortunately the SCM is not a stable model. Since MARITIME a lot of changes have been made at the SCM.

The Ship Common Model is a set of Building Blocks, which are used in the ship product model context. The SCM provides a modelling framework, a set of domain (independent and re-usable) product-structure models that are required for more then one Application Protocol, as well as a set of commonly used constructs or utilities such as those used for configuration control and management

concepts. The goal of the SCM is to contribute to the integration and overall consistency of the Application Reference Models ARM of the different ship AP's.

The SCM is documented in the AP Development Guidelines for Shipbuilding [AP Guide Ship]. These guidelines describing the:

— Building Block approach,

— Modelling guidelines,

— Ship Common Model.

L.3 Modelling guidelines

The modelling guidelines can be seen as extensions to the EXPRESS language defined in Part 11 of STEP. It is specified how to use the EXPRESS constructs for data modelling in the shipbuilding area. These modelling guidelines are also used by the ship common model.

All modelling work for the shipbuilding AP's is done in form of Building Blocks BB. A Building Block is an EXPRESS-based specification that shall be used for the definition of Units of Functionality UoF for an AP. A UoF may include one or several Building Blocks. A Building Block consists of three schemas,

— *import* schema providing an interface for those elements of other Building Blocks to be used by the model schema of this Building Block,

— *export* schema, making available those elements of the model schema intended to be used by other Building Blocks and

— *model* schema defining all the new elements for this Building Block

In addition to the schemas, each Building Block shall have a Building Block header with some administrative information to allow automatic processing by the Building Block e-mail server. The modelling guidelines are related to the Building Block concept. The current guidelines cover the following areas

— Building Block name,

— Size of Building Block,

— Existing Building Blocks,

— Commenting Building Blocks,

— Restrictions on usage of EXPRESS,

— Reference STEP resources,

— Cardinality constraints across Building Blocks,

— Reference functions,

— Reference instances across Application Protocols,

— Importing Building Blocks into Application Protocols and

— Compiling Building Blocks.

Important for the Application Protocol documentation are the guidelines for commenting Building Blocks. These comments are used by tools to produce the clauses 4.2 and 4.3 of an Application Protocol (semi-) automatically.

L.4 Modelling Framework

The modelling framework, which is part of the Ship Common Model, provides the realisation of the general concepts of how to relate things, how to define their properties and how to represent them. The following three EXPRESS based Building Blocks defining the framework:

— definitions,

— generic_product_structures,

— representation_resources.

The framework is a high level approach which split the product model across the main constructs, namely

— items,

— definitions,

— representations

whilst being linked via a number of generic relationships (see Figure 124).

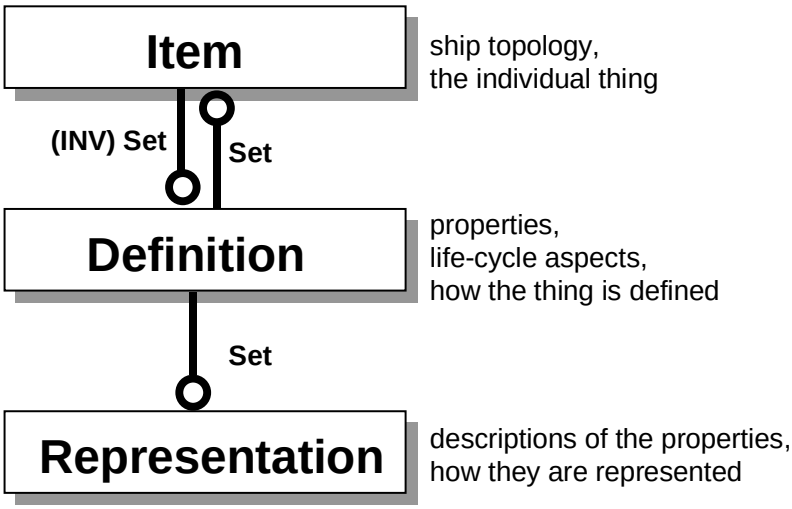


Figure 124 - Modelling framework

Items representing a concept and can be seen as a placeholder. An item is a constant flag during the entire life cycle for a thing. New items in the shipbuilding AP's are introduced by subtyping them from item.

Typical items in the shipbuilding AP's are

— components of the ship (hull, superstructure, deck, rudder, propeller, …),

— equipment (pump, generator, main engine, pipe, …),

— steel structure elements (double bottom, frame, bulkhead, stiffener, …).

The properties of an item are carried by the definitions. A definition must be defined for an item but an item may exist without any definition. The definition can be changed for an item during the life cycle. These life cycle aspects are considered by the life cycle concept (see Figure 125) where different definitions for different life cycle stages are introduced for an item. New definitions in the shipbuilding AP's can be created by subtyping them from definition.

Every property of a concept and therefore, every definition of an item may be described in many different ways. A definition of a moulded form can be represented by a

— offset table representation,

— wireframe representation,

— surface representation,

— spacing position.

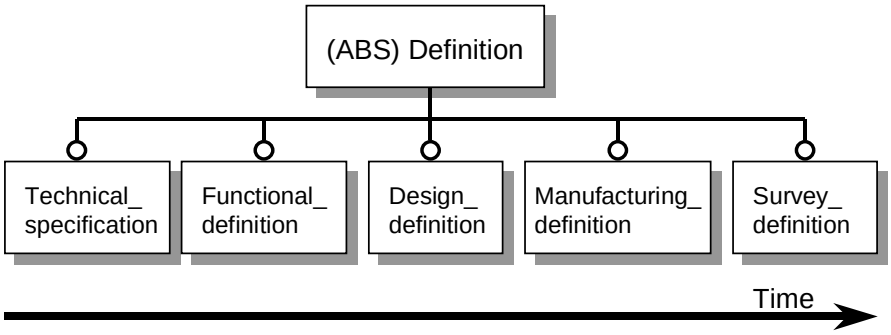


Figure 125 - Life cycle concept

Thus a definition can have different representations, but in some circumstances there can be a definition without representation. New representations in the shipbuilding AP's can be created by subtyping them from representation.

The high level relationships between the main constructs of the modelling framework can be restricted in the subtypes of each AP. This is done by redeclarations of the attributes. For example a moulded form design definition, a subtype of definition is defined for a moulded form, which is a subtype of item (see Figure 126).

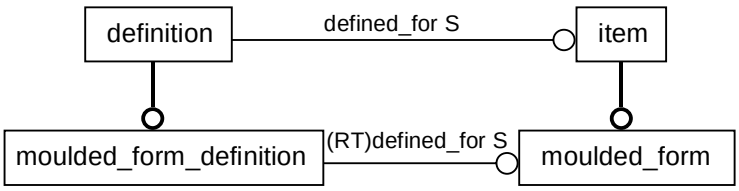


Figure 126 - Redeclaration of attributes

L.5 Domain models

The intention of domain models is to create a new layer of generic elements underneath the main constructs of the modelling framework, with the possibility to restrict the attribute relations using redeclarations. These new elements introduced by the domain models are subtypes of the main constructs of the modelling framework.

The following domain models are part of the Ship Common Model

- Product structure,
- Part,
- Product structure by system,
- Product structure by assembly
- Product structure by space,
- Connectivity.

L.6 Common utilities

Common utilities are EXPRESS based Building Blocks, which are specific for the shipbuilding AP's but on the other side more general, that they can be used in different AP's. They can be described as shipbuilding specific resources.

Common utilities are currently available for

- Ship general characteristics,
- Configuration management,
- Location concepts (global local co-ordinate system, spacing grid),
- Basic geometry and topology,
- Moulded form points,
- Moulded form lines,
- Moulded form surfaces,
- Ships,
- Materials,
- Features,
- Units,
- External reference/ external instance reference,

For more information about the SCM and the modelling guidelines see the AP Development Guidelines for Shipbuilding.

Bibliography

[1] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *Integration Definition for Functional Modeling (IDEF0)* 21st December 1993, National Institute of Standards and Technology Draft Federal Information Processing Standards Publication 183.

[2] Tim Turner: AP Development Guidelines for Shipbuilding, ISO TC184/SC4/WG3 N498, 19th October, 1997

Photographs have been delivered by the following companies:

— Jos.L.Meyer GmbH & Co.

— Kockums Computer Systems Ltd.

— Lloyd's Register of Shipping

— Napa Oy

— Odense Steel Shipyard Ltd.

— Vickers Shipbuilding & Engineering Limited

— Voith Hydro GmbH & Co. KG

— Metallguß Waren GmbH

— NASCO Shipyard

Index

Action_item
AIM diagrams 576
AIM EXPRESS listing ship moulded form type definitions.....392
AIM EXPRESS short listing ship moulded form type definitions.....252
Action_request_item
AIM diagrams 566
AIM EXPRESS listing ship moulded form type definitions.....392
AIM EXPRESS short listing ship moulded form type definitions.....252
Addition_of_moulded_form
application object 80
ARM diagrams 540
mapping table 185
After perpendicular 6
Amidships 6
Appendage_moulded_form_design_parameter
application object 76
ARM diagrams 539
mapping table 168
Applied_action_assignment
AIM diagrams 576
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 260
Applied_action_request_assignment
AIM diagrams 566
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 260
Applied_approval_assignment
AIM diagrams 566
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 261
Applied_classification_assignment
AIM diagrams 575
AIM EXPRESS listing ship moulded form entity definitions 397
AIM EXPRESS short listing ship moulded form entity definitions 253
Applied_date_and_time_assignment
AIM diagrams 577
AIM EXPRESS listing ship moulded form entity definitions 399
AIM EXPRESS short listing ship moulded form entity definitions 261
Applied_document_reference
AIM diagrams 577
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 258
Applied_external_identification_assignment
AIM diagrams 568
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 258
Applied_group_assignment
AIM diagrams 567
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 258
Applied_identification_assignment
AIM diagrams 568
AIM EXPRESS listing ship moulded form entity definitions 398

AIM EXPRESS short listing ship moulded form entity definitions 259
Applied_organization_assignment
AIM diagrams 567
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 259
Applied_person_and_organization_assignment
AIM diagrams 576
AIM EXPRESS listing ship moulded form entity definitions 398
AIM EXPRESS short listing ship moulded form entity definitions 261
Approval_event
application assertion.....125
application object 110
ARM diagrams.....551
mapping table.....231
Approval_history
application assertion.....125
application object 111
ARM diagrams.....551
mapping table.....232
Approval_item
AIM diagrams 566
AIM EXPRESS listing ship moulded form type definitions.....392
AIM EXPRESS short listing ship moulded form type definitions.....252
Bottom_moulded_form_design_parameter
application object 61
ARM diagrams.....538
mapping table.....170
Bulb_moulded_form_design_parameter
application object 62
ARM diagrams.....537
mapping table.....171
Buttock_table
application object 102
ARM diagrams.....544
mapping table.....202
Camber 6
Carrier
application object 39
ARM diagrams.....532
mapping table.....161
Centreline 6
Centreplane 6
Change
application assertion.....125, 126
application object 112
ARM diagrams.....552
mapping table.....235
Change_impact
application assertion.....125, 126
application object 114
ARM diagrams.....552
mapping table.....236
Change_plan
application assertion.....125
application object 114

ARM diagrams 552
mapping table 236
Change_realization
application assertion 126
application object 114
ARM diagrams 552
mapping table 237
Change_request
application assertion 126
application object 113
ARM diagrams 552
mapping table 238
Change_state
application assertion 125, 126
application object 112
ARM diagrams 552
mapping table 240
Check
application assertion 125, 126
application object 114
ARM diagrams 552
mapping table 240
Class
AIM diagrams 565
AIM EXPRESS listing ship moulded form entity definitions 399
AIM EXPRESS short listing ship moulded form entity definitions 262
Class_and_statutory_designation
application assertion 127
application object 48
ARM diagrams 534
mapping table 155
Class_notation
application assertion 127
application object 49
ARM diagrams 534
mapping table 156
Class_parameters
application object 46
ARM diagrams 535
mapping table 157
Classification Society 6
Classification_item
AIM diagrams 575
AIM EXPRESS listing ship moulded form type definitiuons 391
AIM EXPRESS short listing ship moulded form type definitions 250
Compound_representation_item
AIM diagrams 569
AIM EXPRESS listing ship moulded form entity definitions 414
AIM EXPRESS short listing ship moulded form entity definitions 269
Date_and_time_item
AIM diagrams 577
AIM EXPRESS listing ship moulded form type definitions 392
AIM EXPRESS short listing ship moulded form type definitions 253
Deck_moulded_form_design_parameter
application assertion 127

application object 60
ARM diagrams 536
mapping table 182
Definable_object
application assertion 127, 136
application object 104
ARM diagrams 549
mapping table 217
Definition
application assertion 125, 127, 128, 129
application object 108
ARM diagrams 551
mapping table 222
Definition_change_event
application assertion 125
application object 115
ARM diagrams 552
Definition_creation
application assertion 128
application object 115
ARM diagrams 553
Definition_deletion
application assertion 128
application object 116
ARM diagrams 553
Definition_modification
application assertion 128
application object 116
ARM diagrams 553
Definition_relationship
application assertion 128
application object 109
ARM diagrams 551
mapping table 222
Design_definition
application assertion 128
application object 109
ARM diagrams 551
mapping table 222
Displacement_operation
application assertion 129, 130
application object 80
ARM diagrams 540
mapping table 185
Document
application assertion 129
application object 119
ARM diagrams 555
mapping table 228
Document_reference
application assertion 136
application object 120
ARM diagrams 555
mapping table 228
Document_reference_item

AIM diagrams 577
AIM EXPRESS listing ship moulded form type definitions.....391
AIM EXPRESS short listing ship moulded form type definitions.....251
Document_reference_with_address
application object 120
ARM diagrams 555
mapping table 229
Document_usage_constraint
application assertion.....129
application object 119
ARM diagrams 555
mapping table 229
Envisaged_definition_creation
application assertion.....129
application object 115
ARM diagrams 553
Event
application assertion.....129
application object 116
ARM diagrams 553
mapping table 235
Extended_conversion_based_unit
application assertion.....129
application object 123
ARM diagrams 556
mapping table 224
Extended_measure_with_unit
application assertion.....129, 131
application object 123
ARM diagrams 556
mapping table 224
Extended_product_definition_shape
AIM diagrams 568
AIM EXPRESS listing ship moulded form entity definitions 399
AIM EXPRESS short listing ship moulded form entity definitions 262
Extended_product_related_product_category
AIM diagrams 568
AIM EXPRESS listing ship moulded form entity definitions 399
AIM EXPRESS ship moulded form entity definitions 263
Extended_property_definition
AIM diagrams 568
AIM EXPRESS listing ship moulded form entity definitions 399
AIM EXPRESS short listing ship moulded form entity definitions 262
External_identification_item
AIM diagrams 568
AIM EXPRESS listing ship moulded form type definitions.....392
AIM EXPRESS short listing ship moulded form type definitions.....252
External_instance_reference
application assertion.....130, 132
application object 122
ARM diagrams 554
mapping table 227
External_reference
application assertion.....*See*
application object 122

ARM diagrams 554
mapping table 226
External_source
AIM diagrams 565
AIM EXPRESS listing ship moulded form entity definitions 413
AIM EXPRESS short listing ship moulded form entity definitions 270
Floating_position
application assertion 131, 140
application object 84
ARM diagrams 541
mapping table 187
Forward perpendicular 7
Frame_table
application object 102
ARM diagrams 544
mapping table 201
Functional_definition
application object 110
ARM diagrams 551
mapping table 223
General_characteristics_definition
application assertion 130
application object 110
ARM diagrams 551
mapping table 223
Global_axis_placement
application object 98
ARM diagrams 543
mapping table 192
Global_id
application assertion 127, 130
application object 108
ARM diagrams 549
mapping table 221
Group_item
AIM diagrams 567
AIM EXPRESS listing ship moulded form type definitions 391
AIM EXPRESS short listing ship moulded form type definitions 251
Hull_moulded_form_design_parameter
application assertion 130
application object 55
ARM diagrams 537
mapping table 173
Hydrostatic_definition
application assertion 130
application object 79
ARM diagrams 540
mapping table 184
Hydrostatic_position_value
application assertion 130
application object 85
ARM diagrams 541
mapping table 188
Hydrostatic_properties_for_constant_floating_position
application assertion 131

application object 83
ARM diagrams 541
mapping table 187
Hydrostatic_property
application assertion 131
application object 81
ARM diagrams 541
mapping table 186
Hydrostatic_property_value
application assertion 131
application object 85
ARM diagrams 541
mapping table 188
Hydrostatic_scalar_value
application object 85
ARM diagrams 541
mapping table 188
Hydrostatic_table
application assertion 130, 131
application object 80
ARM diagrams 541
mapping table 185
Identification_item
AIM diagrams 568
AIM EXPRESS listing ship moulded form definitions 391
AIM EXPRESS short listing ship moulded form type definitions 251
Item
application assertion 132, 133, 137
application object 105
ARM diagrams 549
mapping table 220
Item_defined_transformation
AIM diagrams 563
AIM EXPRESS listing ship moulded form entity definitions 414
AIM EXPRESS short listing ship moulded form entity definitions 270
Item_relationship
application assertion 132, 133
application object 106
ARM diagrams 549
mapping table 219
Item_structure
application assertion 132, 133
application object 107
ARM diagrams 549
mapping table 219
Knot
application assertion 133
application object 91
ARM diagrams 546
mapping table 212
Local_co_ordinate_system
application object 99
ARM diagrams 543
mapping table 197
Local_co_ordinate_system_with_position_reference

application assertion....133
application object....100
ARM diagrams....544
mapping table....195
Longitudinal_position
application assertion....133
application object....104
ARM diagrams....545
mapping table....205
Longitudinal_table
application assertion....133
application object....101
ARM diagrams....544
mapping table....199
Midship_tumble
application assertion....130
application object....59
ARM diagrams....537
mapping table....175
Moulded_form
application assertion....127, 129, 133, 134, 138
application object....27
ARM diagrams....530
mapping table....147
Moulded_form_characteristics_definition
application assertion....133
application object....54
ARM diagrams....536
mapping table....169
Moulded_form_design_definition
application assertion....134, 138
application object....28
ARM diagrams....531
mapping table....150
Moulded_form_functional_definition
application assertion....134
application object....28
ARM diagrams....530
mapping table....148
Moulded_form_relationship
application assertion....134, 138
application object....27
ARM diagrams....531
mapping table....147
Moulded_form_representation_item
application object....38
ARM diagrams....531
mapping table....153
Moulded_form_representation_relationship
application assertion....134
application object....36
ARM diagrams....531
mapping table....152
Moulded_form_shape_representation
application assertion....134, 135

application object 36
ARM diagrams 531
mapping table 151
Navy_Ship
application object 43
ARM diagrams 532
mapping table 166
Offset_point_table_model
application assertion 135, 137
application object 88
ARM diagrams 547
mapping table 208
Offset_table_shape_representation
application assertion 135
application object 88
ARM diagrams 547
mapping table 208
Organization_item
AIM diagrams 567
AIM EXPRESS listing ship moulded form type definitions 391
AIM EXPRESS short listing ship moulded form type definitions 251
Owner_designation
application assertion 135
application object 53
ARM diagrams 533
mapping table 160
Person_and_organization_item
AIM diagrams 576
AIM EXPRESS listing ship moulded form type definitions 392
AIM EXPRESS short listing ship moulded form type definitions 253
Pitching 7
Planar_symmetry
application assertion 135
application object 37
ARM diagrams 531
mapping table 153
Principal_characteristics
application object 47
ARM diagrams 535
mapping table 154
Product_definition
AIM diagrams 566
AIM EXPRESS listing ship moulded form entity definitions 408
AIM EXPRESS short listing ship moulded form entity definitions 263
Product_definition_relationship
AIM diagrams 567
AIM EXPRESS listing ship moulded form entity definitions 409
AIM EXPRESS short listing ship moulded form entity definitions 270
Propeller_location
application assertion 136
application object 72
ARM diagrams 538
mapping table 176
Propeller_moulded_form_design_parameter
application assertion 136, 141

application object 64
ARM diagrams 538
mapping table 177
Property_definition
AIM diagrams 566
AIM EXPRESS listing ship moulded form entity definitions 409
AIM EXPRESS short listing ship moulded form entity definitions 264
Property_definition_representation
AIM diagrams 574
AIM EXPRESS listing ship moulded form entity definitions 410
AIM EXPRESS short listing ship moulded form entity definitions 266
Regulation
application assertion 127, 136
application object 50
ARM diagrams 534
mapping table 156
Representation
AIM diagrams 574
AIM EXPRESS listing ship moulded form entity definitions 414
AIM EXPRESS short listing ship moulded form entity definitions 267
Research_Ship
application object 43
ARM diagrams 532
mapping table 165
Revision
application assertion 136
application object 117
ARM diagrams 553
mapping table 232
Revision_with_context
application assertion 136
application object 117
ARM diagrams 553
mapping table 233
Rolling 7
Rotational_symmetry
application assertion 137
application object 37
ARM diagrams 531
mapping table 153
Rudder_moulded_form_design_parameter
application assertion 137
application object 73
ARM diagrams 539
mapping table 179
Section_of_offset_point_table
application assertion 135, 137
application object 89
ARM diagrams 547
mapping table 209
Sheer 7
Ship
application assertion 130, 132, 137, 139
application object 107
ARM diagrams 550

mapping table 220
Ship moulded form entity definitions 253
Ship moulded form imported entity modifications 263
Ship moulded form type definitions....250
Ship moulded function definitions 386
Ship moulded rule definitions 271
Ship_curve
application assertion....133, 137, 138
application object 93
ARM diagrams 548
mapping table 214
Ship_curve_segment
application assertion....137
application object 91
ARM diagrams 546
mapping table 211
Ship_curve_with_spacing_position
application assertion....138
application object 97
ARM diagrams 548
mapping table 215
Ship_designation
application object 51
ARM diagrams 533
mapping table 158
Ship_moulded_form
application assertion....130, 138, 139
application object 26
ARM diagrams 530
mapping table 146
Ship_moulded_form_revision
application assertion....138
application object 36
ARM diagrams 531
mapping table 148
Ship_moulded_form_wireframe_model
application assertion....138
application object 90
ARM diagrams 546
mapping table 210
Ship_overall_dimensions
application assertion....138
application object 54
ARM diagrams 536
mapping table 175
Ship_point
application assertion....137, 139
application object 92
ARM diagrams 547
mapping table 214
Ship_surface
application assertion....139
application object 97
ARM diagrams 548
mapping table 216

ShipType
application assertion....139
application object....38
ARM diagrams....532
mapping table....160
Shipyard_designation
application assertion....139
application object....52
ARM diagrams....533
mapping table....159
Spacing_position
application assertion....138, 139
application object....102
ARM diagrams....544
mapping table....204
Spacing_position_with_offset
application assertion....139
application object....104
ARM diagrams....545
mapping table....206
Spacing_table
application assertion....139
application object....101
ARM diagrams....544
mapping table....198
Speed_measure_with_unit
application object....124
ARM diagrams....556
mapping table....224
Speed_unit
application object....123
ARM diagrams....556
mapping table....225
Stability_definition
application assertion....139
application object....85
ARM diagrams....542
mapping table....189
Stability_properties_for_one_floating_position
application assertion....140
application object....87
ARM diagrams....542
mapping table....190
Stability_property
application assertion....140
application object....87
ARM diagrams....542
mapping table....190
Stability_table
application assertion....139, 140
application object....86
ARM diagrams....542
mapping table....189
Station_table
application object....102

ARM diagrams 544
mapping table 200
Stem 7
Subtraction_of_moulded_form
application object 80
ARM diagrams 540
mapping table 185
Summer load waterline 7
Surface_shape_representation
application assertion 141
application object 92
ARM diagrams 545
mapping table 213
Symmetry
application assertion 135
application object 37
ARM diagrams 531
mapping table 153
Thruster_moulded_form_design_parameter
application assertion 141
application object 75
ARM diagrams 539
mapping table 181
Transversal_position
application assertion 141
application object 104
ARM diagrams 545
mapping table 205
Transversal_table
application assertion 141
application object 102
ARM diagrams 544
mapping table 201, 202
Universal_resource_locator
application object 120
ARM diagrams 554
mapping table 226
UoF
basic_geometry 12
configuration_management 12
date_time_resources 13
definitions 13
external_references 14
hydrostatics 14
items 15
location_concepts 15
measures_resources 16
offset_table_representations 17
organisation_resources 17
representations 18
shape 18
ship_design_parameter 22
ship_general_characteristics 23
ship_measures 24
ship_moulded_forms 24

surface_representations 25
wireframe_representations 25
Version_history
application assertion 141
application object 118
ARM diagrams 553
mapping table 233
Version_relationship
application assertion 141, 142
application object 118
ARM diagrams 553
mapping table 234
Versionable_object
application assertion 136, 141, 142
application object 105
ARM diagrams 549
mapping table 244
Vertical_position
application assertion 142
application object 104
ARM diagrams 545
mapping table 206
Vertical_table
application assertion 142
application object 102
ARM diagrams 544
mapping table 204
Waterline_table
application object 102
ARM diagrams 544
mapping table 203
Weather deck 7
Wireframe_shape_representation
application assertion 142
application object 90
ARM diagrams 546
mapping table 210
Working_Ship
application object 41
ARM diagrams 532
mapping table 163
Yawing 7