On capturing information requirements in process specifications

Edward J. Barkmeyer and Peter Denno[1]

Keywords: process modelling, information modelling, information requirements, ontologies

Background

Process and information modelling

Business process modelling has been in vogue for nearly 20 years, and its origins are much older than that. The growth in the last 20 years can be attributed primarily to inexpensive and powerful computer graphics technologies, which enabled rapid display and modification of complex diagrams. Software process modelling goes back to the flowchart era of the 1950s and 1960s, and reached its high point in the Structured Analysis and Design era (1970-85).

Structured analysis taught that one defined information systems by modelling the to-be business process at a high level and refining each high level activity to its component activities, flows and decisions, and repeating this process until the "activities" could be reduced to a few machine instructions. But information requirements were described only at a very high level. In a language of the time, like IDEF0 [1], the basic idea of "information" is essentially a "document" idea. A body of information is characterised by a name and an overall description – the individual information units are not given at all. And this view of integrating information into process specifications is still very much alive in proprietary

[1] National Institute of Standards and Technology, 100 Bureau Drive, MS 8263, Gaithersburg, MD 20899-8263, USA
edward.barkmeyer@nist.gov
peter.denno@nist.gov

business process specification languages[2], such as ARIS [2] and METIS, and in the recent Business Process Modeling Notation (BPMN) standard [3].

Since the early 1980s, the approach to modelling information has concentrated on the business entity as the cluster point for information units. The information model is a collection of "entities" or "objects" that are connected by "relationships", and each entity and relationship is represented by a collection of information units ("attributes") that describe its current state. Information modelling languages like UML class diagrams [4], IDEF1-X [5], ORM [6], and EXPRESS [7] all have exactly this structure. They also all support the idea of classes and subclasses of entity instances, which allow the collection of information units associated with an entity instance to be characterised by a template. Models of this kind lead directly to the construction of relational or object-oriented database schemas that capture these information units in structures that correspond to the classes. The information modelling approach is important, because it captures an important element of the rationale for information – description of the entities and relationships that are relevant to a business process.

But these information concepts have rarely been coupled with process models. And, as a consequence, we see many interchange specifications that document the detailed information exchange elements and the public business process that justifies the specification, with no clear association between them. Because it is not possible, under these circumstances, to trace from an element of the exchanged content to the actual need that motivated its inclusion in the exchange, such interchange specifications cannot play the important role of providing complementary detail to the actual, specific, business process model. As a consequence, agility in the tasks of addressing new business needs, re-engineering, and systems integration is lost.

Message modelling

Part of the problem is that the information modelling approach was originally developed for the design of databases – everything that a complex of software applications may need to know in order to support many business processes. The Open Applications Group "business object document" definitions [8], and the EDIFACT [9] electronic business transaction designs, have the same flavour: there is a "component" that corresponds to an entity, and it contains a data field for every information unit one might want to convey about the entity. The information about an entity includes its relationships to other entities, and the content of such a relationship field is the "component" that describes that entity. But because one may want to convey only a small set of information about any given entity, all of the fields are optional. And the STEP "Integrated Resource Models" [10] are structurally the reverse, but effectively identical: There is a large set of entities that have nearly no attributes. Every attribute and relationship is a separate model element, and a given exchange specification explicitly collects the needed entities and attribute/relationship model elements. The culmination of these approaches to

[2] References to proprietary products are included in this paper solely to identify the tools actually used in the industrial applications. This identification does not imply any recommendation or endorsement by NIST, as to the suitability of the product for the purpose

transaction design is the UN/CEFACT Core Component Technology [11], which is a very rigourous way of doing exactly the same thing to define the components of XML-based messages.

These "technologies" don't capture the required information for a given business process, nor do they model the information exchanged in a given interaction. They depend on a later agreement by a small community or a pair of business partners to define the detailed requirements for the messages in a given business process. The real OAG message definitions depend on "message implementation guidelines" (MIGs), captured formally in a proprietary language (GEFEG), that define the set of entities and fields that are to be included in a message for a given purpose in a given process. The STEP messages tend to be large information sets, but they also have a language for defining the sets of entities, attributes and relationships to be included in an exchange file ("application protocol"), and they refer to such a specification as a "mapping table". The UN/CEFACT activities are still working on languages for defining restrictions and collections of components, including the Core Component Message Assembly specification [12] and the Universal Context Mechanism (UCM) [13], among others.

It is important to note that all of these "assembly technologies" are used not only to eliminate unused information units and relationships, but also to refine or specialise the entity or information unit in the particular usage (context). That is, the assembly language introduces distinguishing characteristics of roles and subclasses that are not explicitly modelled. And in settling on the nebulous “context” rather than discriminating between process roles, relationship roles and distinctions among domain objects, these technologies tend to *hide* business knowledge rather than capture it.

The problem is that these mechanisms for transaction design assume that the *need* for specific information units is somehow externally specified. That need is derived from characteristics of the business process that necessitates the flow of information. But the business process model itself does not capture that.

The UN/CEFACT Modeling Methodology (UMM) [14] is an attempt to rectify this problem. It describes a Business Process View (BPV) that interrelates process steps in a UML Activity Diagram with states of objects that are shared by collaborating partners in a joint activity. Because it highlights objects, rather than messages, as a focal point of the transaction, it is clearly a step forward. However, "state" in the BPV is an abstraction over unspecified properties of the business entity (and perhaps other entities). It does not identify the properties that are relevant, and many properties of the entity may be may be important to other joint processes, but irrelevant to the activity at hand. Further, the BPV view of the shared object does not call out information elements that identify individuals; it makes the assumption that there is always only one object of a kind that is the subject of a joint activity, and it is unclear how that object relates to other objects in the shared viewpoint of the partners. This approach is good, but inadequate.

Exchange standards

As we have seen in working with the AIAG Inventory Visibility and Interoperability specification [15], the relationship between the process activities

and the requirements for the information units is not well documented, e.g. a description of a Shipment. Similarly, a STEP Application Protocol, such as Product life cycle support (PLCS) [16], contains many pages of documentation of the process and its relationship to information, but the process is abstract, and the relationships are described only at the level of documents, e.g., a document describing the structure of a part from a fabrication viewpoint. In each case, the process specification describes the information needs at a very high level and another specification documents the information requirements in detail, typically without one word about process elements.

These specifications are meant to be industry standards, and they are intended only to *enable* exchanges of information, rather than to characterise those exchanges in any detailed way. In that sense, they have the same purpose as the database – to support a great many distinct processes. In the STEP case, the purpose is to support a set of engineering processes that need much of the same information, and the intent is primarily to specify the sets of information that must be published by the engineering support tools that capture that information. In the OAG and EDIFACT cases, the intent is to define an abstract public process that is realised in practice by many somewhat different private processes. The standard information set for each message, therefore, comprises all of the components/entities that any of these realisations might actually need.

Proposal

Whether the real message/document contents are defined in the standard or by the users, the problem remains that the only record of the decisions made about information requirements are the specifications for the data elements to be included in a "software interface" – a message, a document, a procedure call. If the process determines the information requirements, that relationship is not specified. As a modelling community, we do not have a methodology for capturing these relationships, and we do not have a language for expressing them.

In our view, a methodology for specifying the needed information flows in a joint business process involves developing three major components:

- a "reference ontology" for the business entities
- a formal specification for the joint process
- a binding between the process elements and the business entities

The reference ontology

"Ontology" is the term currently in vogue for a formal information model. An ontology "class" is an "entity", a "datatype property" is an "attribute", and an "object property" is a "relationship". Ontology languages are distinguished from previous information modelling languages by defining the meaning of the model elements in mathematical logic. Previous work [17] has assigned logic-based semantics to information modelling languages after the fact, but those semantics were not necessarily supported by tools and practitioners. There is a sometimes

important distinction in the capture of relationships: information models usually capture relationships as owned by a particular entity/class, while logic models regard a property as the same level of concept as a class – the "domain" of a property may or may not be constrained to a particular class.

The real contribution of ontologies is to capture the properties that define a classification – what properties do individuals of this kind always have that instances of the parent broader classification do not necessarily have? Information modelling languages allow one to state “necessary conditions” for membership, e.g., "if x is an instance of class C, x has (or may have) property P." But ontology modelling languages also allow one to state “sufficient conditions” for membership, e.g., "if x is an instance of parent class Y and has property Z, then x is an instance of class C." Information modelling and object modelling methodologies suggest that there should be such characteristics for a classification, but the modelling languages generally cannot capture them. Best practice in ontology definition demands that these characteristics be explicitly modelled. This enables clear agreement on the subjects of a transaction or the interpretation of an information unit that describes them.

The scope of the reference ontology has to be at least everything that is relevant to the process and is shared between participants, and may be everything relevant to a set of related processes. In short, the scope of the reference ontology is the same as a large part of the conceptual database schema, or some part of the EDIFACT component dictionary. But unlike the conceptual schema, the reference ontology need not contain descriptions of entities, relationships or information units that have no public image. Concepts that appear only inside the "black box" that is the private process of a participant need not be part of the reference ontology.

Since those conceptual schemas and component dictionaries already exist in some form, the requirement for the future is only to formalise those that are not formalised and to explicitly capture the subclass characterisations. But a major requirement may be to reconcile the disparate ontologies of the separate participants in a community that would make them partners.

It would be helpful if these ontologies were based on some accepted "upper domain ontology" – a set of terms, classifications, and information units that everyone understands. The existence of a well-defined set of high-level classifications allows almost every domain classification to be well-defined in terms of characteristics, as indicated above. And conversely, the absence of well-defined high-level classifications means that a useful domain ontology has many "primitive classes" – intuitive classifications that have no formal parents or distinguishing characteristics.

But we can only base the formalisation of a conceptual schema on an upper ontology that exists, is accepted, and is useful in the business domain. And at this moment, such ontologies really only exist in some areas of medicine [18]. In many business domains, there is an accepted collection of high-level generalities, but not widespread agreement on the use of those as classifications for individual entities. Ultimately, the reference ontology for any given set of business transactions only needs to reflect agreement of the participants, and if that agreement involves many primitive classifications with common intuitive understanding, then it does.

It seems to be a goal of a number of standardisation projects, notably ISO 19440 [19], PLCS, UCM, and DODAF/MODAF [20], to foster the development of accepted "upper domain ontologies" for particular industries, while allowing for local extensions and variants, wherever needed. This approach has some promise for supporting the business reality.

The formal specification for the joint process

The joint process specification is simply a formal specification for the process that identifies all the interactions (activities) that involve exchange, publication, or retrieval of (shared) information. Modes of communication relevant to this specification include publication/retrieval as well as directed peer-to-peer. Thus it includes asynchronous and indirect communications as well as messaging. The intent here is to capture the required communications, whatever form they might take.

It is not uncommon for business process models to capture these interactions, and a number of process modelling languages (e.g., GRAI [21], ARIS, BPMN) support the identification of messages flowing between activities and flows of other information artefacts. The thrust here is to ensure that the specification is complete in this regard. The joint process, and the public view of the supporting private process elements is the focus of the model.

In general, the joint process model will have a "decomposition". The high-level joint process will involve joint activities, and each joint activity can be described as a process that involves joint and separate activities. And this decomposition will continue until all that remains is separate activities of the participating agents, some of which involve information exchanges with other participating agents. Figure 1 depicts a joint Customer/Supplier process for a Vendor-Managed Inventory supply arrangement using the BPMN language.

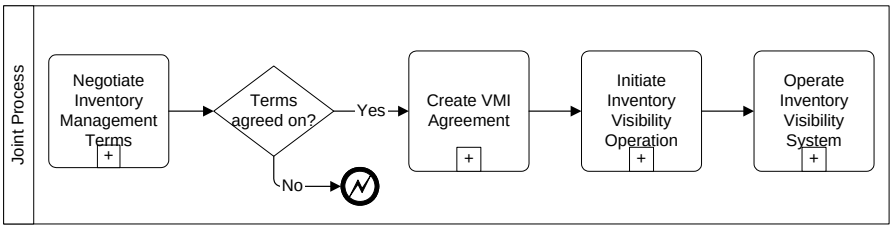

Figure 1: BPMN diagram of joint Vendor-Managed Inventory Process

Figure 2 depicts the decomposition of the first joint task into the cooperating public processes of the Customer and Supplier. The dotted arrows denote communications between the participants, and the circled envelopes denote points at which the process waits for such a communication.

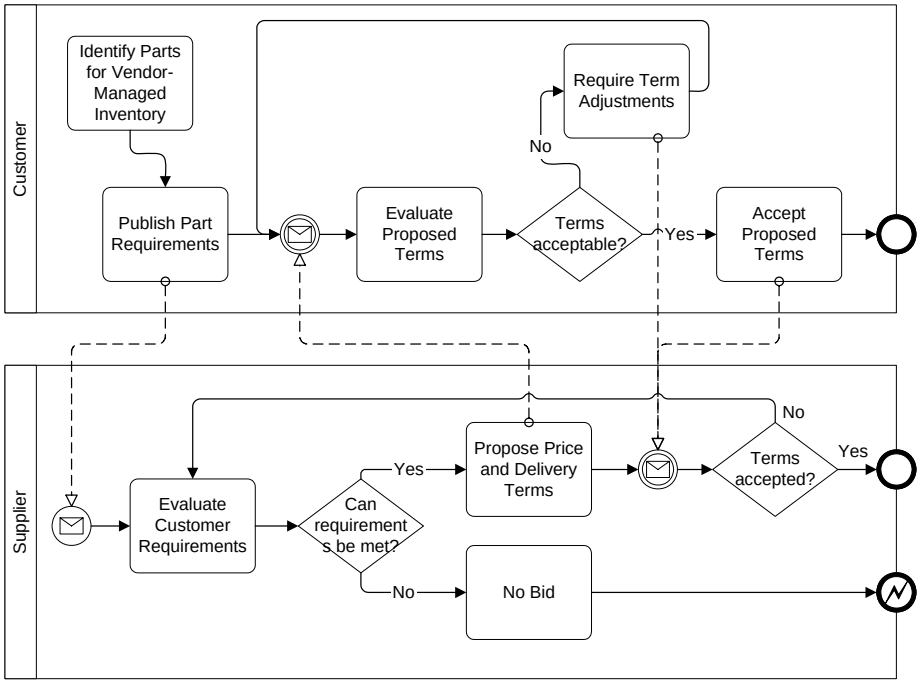

Figure 2: Negotiation task expanded to Customer and Supplier processes

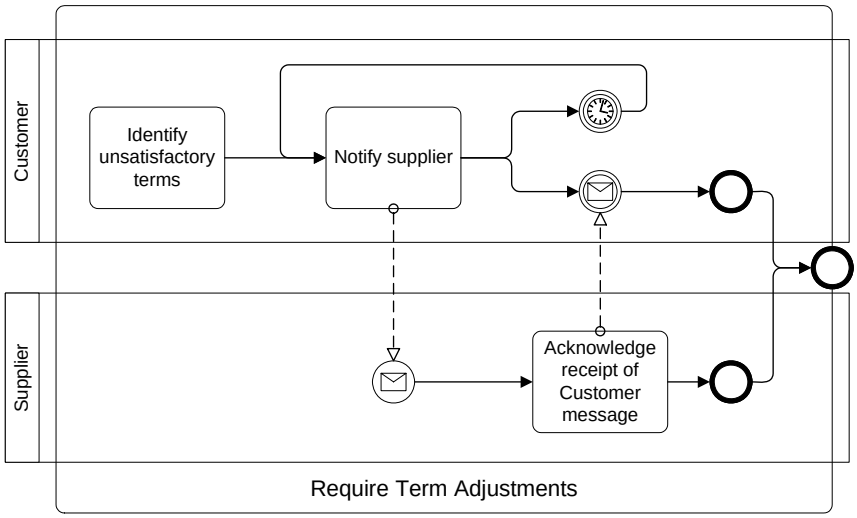

Figure 3 Expansion of the Require Term Adjustments task

In some cases, a higher-level view of the process may identify an explicit information exchange that itself expands into a sequence of component information exchanges. Figure 3 depicts an expansion of the task Require Term Adjustments (from Figure 2) into a transmission and an acknowledgement. This low level of

detail is only necessary when the purpose is to define explicit messages and their contents.

In the recent past, several modelling languages have added features or interpretations to address the capture of participant interactions in detail. And this is in part a response to demand in the software community for "business process models" that identify specific communications among software agents, such as "electronic business" messages and webservice invocations. The more general problem is to capture the required interactions of business agents at every level of design of their joint activity, and to capture at each level the required communications among the agents at the corresponding level of abstraction.

The association of the information requirements

The information requirements association is a formal specification that associates the activities in the process with the business entities in such a way as to characterise the relationship of the activity to the *state* of the business entity. From this, it is possible to derive the information units the activity requires, and thus the information that must be provided via the communications paths.

In associating an activity with the state of an entity, there are exactly six alternatives:

- using one or more properties of an entity instance or relationship
- changing one or more properties of an entity instance or relationship
- creating an entity instance
- creating a relationship (instance) between two entity instances, where one of them is usually considered to be the "subject" (or "domain") and the other the "object" (or "range") of the relationship.
- destroying an entity instance
- destroying a relationship instance

These actions correlate to the primitive operations of a database interface – Retrieve, Insert, Update, Delete – and the database versions of these operations have been a part of "data transaction analysis" for 25 years, and they are explicitly documented in the "information viewpoint" of a system according to the Reference Model for Open Distributed Processes (RMODP) [22]. The OPAL language [23], which is intended to provide an upper ontology of business process for the development of business interaction models, provides exactly these operations as characteristics of a BusinessActor, which is a model of a process role.

Of course, these operations often also correlate to events in the world described by the information. Artificial entities may in fact be created and destroyed, but real entities and relationships are not "created" and "destroyed"; rather the facts they represent become important to the business process or become unimportant or cease to hold.

In a certain sense, only one of the above operations is important in defining the information flow requirements. When an agent, in conducting a modelled activity, *uses* a property of an entity or relationship, there is a demonstrated *information requirement*. When the objective is to capture information requirements associated with the process, this is the point at which that happens. No further model details are necessary to achieve that objective.

That said, all of the other operations above create *events*, or knowledge of events – they change the world in which other activities are taking place. And knowledge of events of a particular kind may constitute an information requirement for an activity as well. Many process modelling languages can represent external events and the activities that are the response to them, so this is an area in which at least the high-level information requirement is often modelled in the process specification.

Like the process model itself, the information requirement can have a fine-grained form, down to the property that is an information unit, or to an aspect of the entity (some conceptual set of properties that relate to a particular use or behaviour), or simply to the entity itself. This is where the traditional EDI approach, and the capabilities of the UML Activity model (used in the UMM), become inadequate – they cannot address any resolution finer than the entity, when in many cases only a few modelled properties are used.

When the objective is to define the communications in detail, the business process engineering activity involves a further "analysis and design" process. The use of a property constitutes an information requirement, whether the agent that uses it has that information or not; but when the agent does not have that information, the requirement becomes a need for an information *flow* – the requirement must be satisfied by some transmission of information to that agent. There is an associated analytical process that identifies a *source* for the needed information, and a design process that chooses a *mechanism* for its transmission. Ultimately, the design process will produce a refinement of the interactions of the two agents that specifies the details of the information flow. In such flows, a reference to an entity instance implies use of some set of properties ("keys") that are used to identify the individuals. Across business partners, agreements on the use of identifying keys for shared and mirrored entities is critical to the design.

All of this engineering is necessary to develop the design for messages and database schemas and the like. This engineering activity refines the model "activity role X uses property P of entity E" into separate actions of the participants, such as: "Participant Y provides property P of entity E" and "participant X receives/fetches property P of entity E". And, of course, the engineering optimisation rolls up a group of such information transfer requirements into a single message, service invocation or database transaction. But none of this engineering process is necessary to capturing the *requirement* and associating it with the process model. The information requirement arises directly from the activities in the business process. Once identified, it can be used in characterising the contents of communications among the participants.

In several of the UML process specification languages it is possible to specify detailed flows of information between agents as uses of "object interfaces". It is important to recognise that the interface specification documents some of the information needed by the agent to perform the documented task, and some of the information produced by the agent in performing it. But such models reflect decisions about how information will be provided, and do not identify information the agent needs and is expected to have available in some "encapsulated" way. And they depend on further expansion of the design of the agent to define information requirements the agent fills by interacting as "client" with other agents,

such as data repositories. Such models are not intended to be information requirements models, but rather designs for meeting the (previously identified) information requirements with a chosen organisational model (component architecture).

What we propose is that each activity in a process have associated "user flows", each of which identifies an entity from which the information originates and the properties of that entity that represent the needed information. Similarly, each activity may have associated "provider flows", each of which identifies an entity whose state is altered by the activity, and the properties of that entity that are affected.

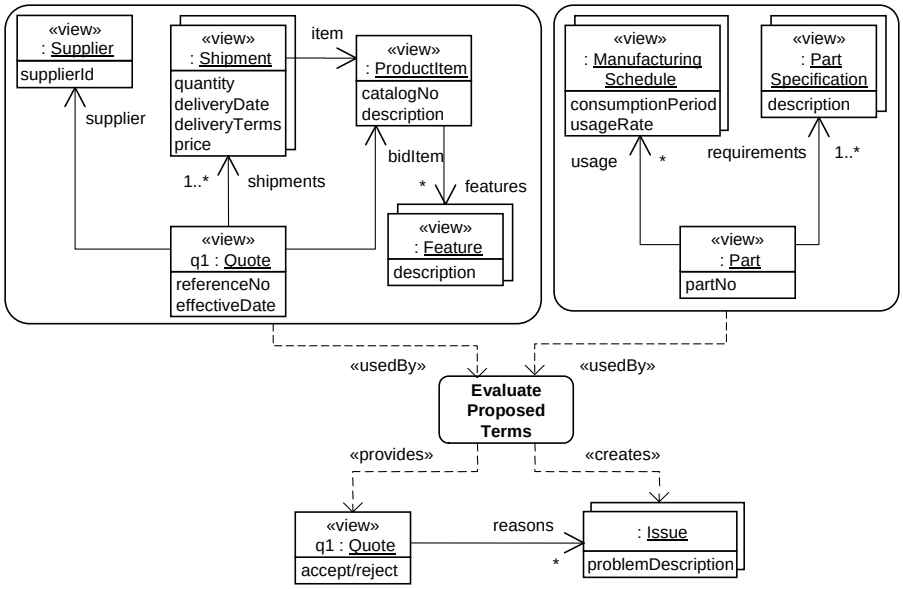

Figure 4: Consumer and Producer Flows for "Evaluate Proposed Terms"

Figure 4 shows the user and provider flows for the Evaluate Proposed Terms activity. This is an attempt to exemplify what is needed, and should not be taken as a proposed notation. The diagram depicts only the properties from the ontology that are involved in each flow, indicated by the stereotype «view» for each entity involved. In practice, the ontology will include other properties for these entities, and some means of specifying the selection will be needed. In addition, some of the properties that are used or altered are actually properties of entities reached by some relationship path from a "subject" entity of interest. Finally, note that the Quote being altered in the provider flow is the same as the one being used in the user flow, and it must be identified by supplier and referenceNo, although these properties are not modified. So, while the proposed concepts are simple, the actual representations of user and provider flows may be complex. This requires further research.

Summary

The above demonstrates that there is a shortcoming in our current modelling technologies and practices – we don't have a discipline, or a language, that relates the capture of information requirements to the process activities that have those requirements. We capture at most the idea that the activity relates to a particular business entity.

As a consequence from a business modelling point of view, we are not documenting the rationale for information flows. And therefore, we find it necessary to do broad analysis and re-design in attempting to respond to changes in the business processes or changes in the information requirements.

As a consequence from a technical point of view, we have no basis for automated reasoning about exchanges of information. We cannot reconcile disagreements about the contents of messages between divers business partners, because we have no basis for determining what the information requirements are. We have only the fiat of each partner as to the required content of a message, which is usually based on the nominal content of a standard message in which the given process uses some of the information. There is no process-based justification for the information flows or the message content.

We propose an approach to documenting information requirements as they arise in the joint business process. And we suggest that what must be added to a process model to support this is conceptually simple, but its representation has ramifications that require more research.

We also suggest that this approach is a way of formally bringing together process modelling and information modelling in a way that capitalises on the strengths of both disciplines.

References

[1] Federal Information Processing Standard Publication 183, *Integration Definition for Functional Modeling (IDEF0)*, National Institute of Standards and Technology, December 1993, available from National Technical Information Service, U.S. Department of Commerce, Springfield, VA.

[2] Scheer, A. W., *Business Process Engineering: Reference Models for Industrial Enterprises*, (Second Edition) Springer, Berlin, 1997.

[3] Object Management Group, *Business Process Modeling Notation Specification*, OMG Final Adopted Specification, February, 2006, available at: http://www.omg.org/cgi-bin/doc?dtc/2006-02-01 [cited 1 Nov 2006].

[4] Object Management Group, *Unified Modeling Language (UML)*, version 2.0, July 2005. Available from http://www.omg.org/technology/documents/formal/uml.htm [cited 1 Nov 2006]

[5] Federal Information Processing Standard Publication 184, *Integration Definition for Information Modeling (IDEF1-X)*, National Institute of Standards and Technology, December, 1993, available from National Technical Information Service, U.S. Department of Commerce, Springfield, VA.

[6] Halpin, T.A., *Object-Role Modeling: An Overview*, Springer, San Francisco, 2000.

[7] International Organization for Standardization, ISO 10303-11:2004, Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual, Geneva, 2004.
[8] Open Applications Group, *Open Applications Group Integration Specification (OAGIS)* v9.0. Available via: http://www.openapplications.org/downloads/oagidownloads.htm.
[9] United Nations Economic Commission for Europe (UNECE), UN/EDIFACT Standard Directories, v6.0, 2006, available at: http://www.unece.org/trade/untdid/directories.htm,.
[10] International Organization for Standardization, ISO 10303-1:1994, Industrial automation systems and integration -- Product data representation and exchange -- Part 1: Overview and fundamental principles, Geneva, 1994.
[11] International Organization for Standardization (ISO), ISO WD 15000-5:2006 Core Components Technical Specification 2nd Edition, March 31, 2006 Working Draft, 2006.
[12] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) Core Components Message Assembly Technical Specification, Working Draft 0.4,July, 2006.
[13] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), Unified Context Methodology Project Proposal, March 23, 2006.
[14] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) UN/CEFACT Modeling Methodology Foundation Module Version 1.0 – Technical Specification, 2006-10-06
[15] Automotive Industry Action Group, Publication I-1, *Inventory Visibility and Interoperability– Proof of Concept – Phase I Project Summary*, Southfield, Michigan, March, 2005.
[16] International Organization for Standardization, ISO 10303-239:2005, Industrial automation systems and integration -- Product data representation and exchange -- Part 239: Application protocol: Product life cycle support, Geneva, 2005.
[17] Halpin, Terry A., *A Logical Analysis of Information Systems: Static Aspects of the Data-oriented Perspective*, Department of Computer Science, University of Queensland, 1989.
[18] U.S. National Library of Medicine, Unified Medical Language System Fact Sheet, available at: http://www.nlm.nih.gov/pubs/factsheets/umls.html [cited 1 Nov 2006]
[19] International Organization for Standardization, ISO FDIS 19440 Enterprise integration – Constructs for enterprise modelling, Geneva, 2006.
[20] U.S. Department of Defense Architecture Framework Working Group, *DOD Architecture Framework Volume 1.0*, September, 2003, available at: http://www.enterprise-architecture.info/Images/ Defence%20C4ISR/Enterprise%20Architecture%20Defense.htm [cited 1 Nov 2006]/
[21] G. Doumeingts , D. Chen , B. Vallespir , P. Fenie - *GIM (GRAI Integrated Methodology) and its evolutions - A methodology to design and specify Advanced Manufacturing Systems*. In JSPE-IFIP 5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing, Tokyo, Japan, 8-10 November 1993, 16 p.
[22] International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 10746-2:1996, Information technology – Open Distributed Processing – Reference model: Foundations, Geneva, 1996
[23] Missikoff, M., Taglino, F., ATHENA Project: Knowledge Support and Semantic Mediation Solutions, Part B, the ATHOS Specifications, March, 2005, to be published.