NIST Industry Forum

08 May 2018

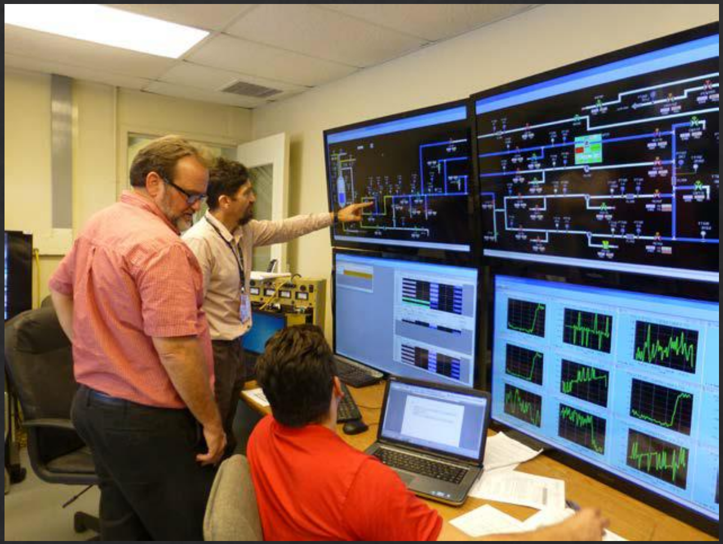
Crafting Intelligent Systems Management Using Requirements-Driven Design

Mark Walker D2K Technologies

mark.walker@d2ktech.com

Agenda

- Intro
- **PHM Overview**
- **Requirements Driven Design**
  - Definitions
  - Generic Approach
- **Re-usable OO Platforms**
  - Overview
  - Architecture



Who Are We?

- A Solutions company established in 2014, D2K utilizes:
  - reliability centered design methodologies
  - state of the art **OO AI** software development platforms (we love reuse!)
  - agile software engineering for on-time delivery of validated software solutions

- Focus: to leverage system model-based reasoning for delivering "Situation Aware" software. SA software is "thinking" software that encapsulates insight and understanding regarding operation, availability, uncertainty, and adaptation.

- …software that can intelligently and autonomously monitor, control, emulate, execute, or optimize actions that will successfully ensure safe, timely, and dependable results.

Prognostics and Health Management (PHM)

PHM Systems are evolving to meet higher expectations

- **What should PHM Systems do?**
  - Determination of Health and its impact on system functions
  - Monitor early warning of incipient failures
  - Predictions of Remaining Useful Life
  - Leveraging of advanced "reasoners"
    - Signal processing for event detection
    - Algorithms for event correlation and sensor fusion
    - Expert Systems and rule-based architectures
    - Advanced neural and statistical classifiers
    - Real-time state estimators
    - Model-based Reasoning
  - Supervisory-level intelligence / logic
  - Estimation and understanding of system state within operational context
  - Decision support to assist operators in maintaining operational availability
  - Optimize scheduling of maintenance and corrective actions according to the principals of condition-based maintenance

Prognostics and Health Management (PHM)

- **How have PHM Systems performed?**
  - Expensive
    - Takes too long to develop and deploy
  - Often ill specified
    - Limited access to existing design data
    - Incomplete (or non-existing) design data
  - Often an afterthought - considered very late in design cycle
    - Often reduced in scope
    - Or involving small incremental improvements to legacy systems
    - Or eliminated altogether
  - Excruciating test and validation cycles
    - How to qualify PHM system?
    - No false positives / no false negatives
    - Test and validation using Simulation vs. historical data vs. supervised learning
  - Questionable performance
    - Is system availability increased (downtime minimized)?
    - Is MTTR decreased?
    - Are operators better equipped and informed?
    - Are overall lifecycle costs reduced?

System Management Design Considerations

- Early as possible derivation of requirements
- Design based on functional requirements and the mitigation of failures (Behavior Driven Design)
- Need to link failures to detectable events across subsystems, and diagnosis to maintenance and corrective actions
- Design should identify necessary instrumentation (and consequences of inadequate instrumentation)
- Design should consider reasoning over systems, subsystems, predictive models, usage, operational regimes, real-time and historical data – within operational context
- Design should offer immediate advantages for life-cycle management

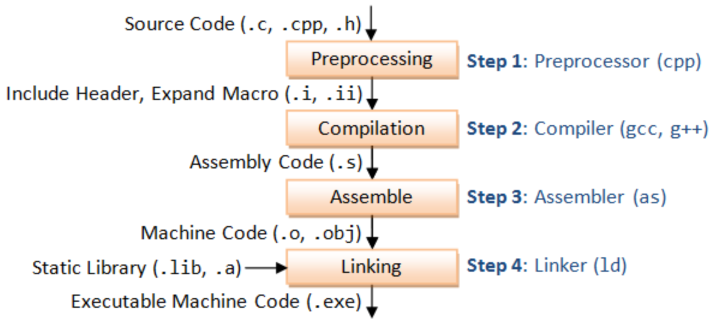
Output of Design Methodology

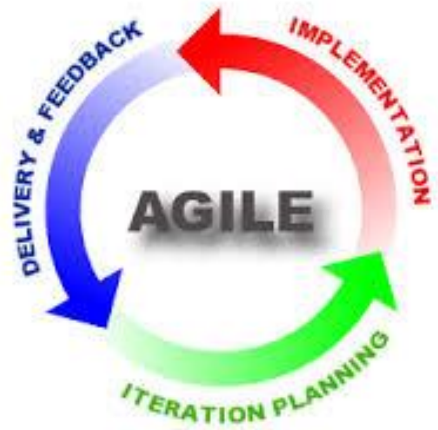
- Crisp set of System Management requirements according to operational context, functional requirements, and mission objectives
- Preliminary definition of critical failure modes, associated instrumentation and algorithms required to detect them, and downstream consequences (as well as the intra-subsystem event propagations that drive them)
- Baseline system object model required for reasoning
- Baseline fault models for diagnosis and prognostics
- Simulation and initial validation of diagnostic approaches and understanding of underlying event propagation

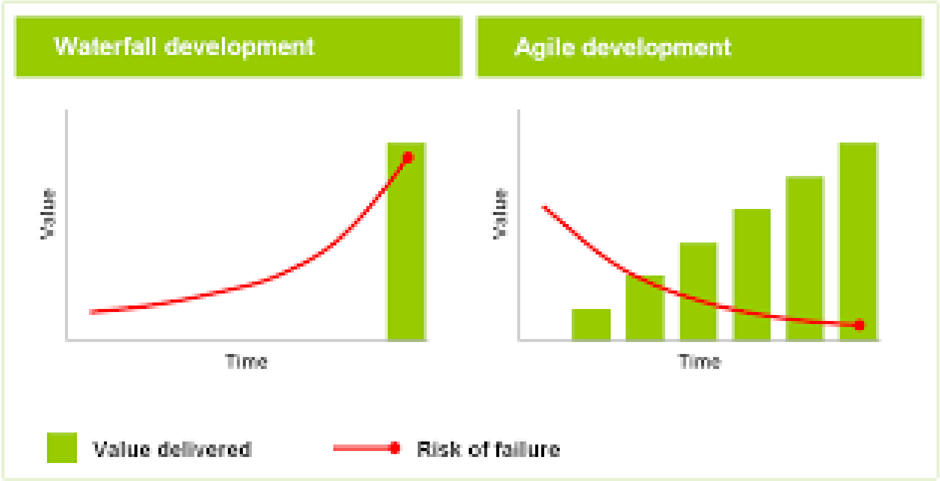
Quest for Software Quality

- **Test Driven Design (TDD)**
  - Write a test that fails
  - Code until it passes
  - Refactor (re-coding if it breaks)
- **Behavior Driven Design (BDD)**
  - "BDD is about implementing an application by describing its behavior from the perspective of its stakeholders"
  - Requirements as User Stories
  - Pull vs. Push based
- **Automated Testing using philosophy of jUnit, TestNG (example tools)**
  - Automated Report Generation
  - Tests follow system through life-cycle

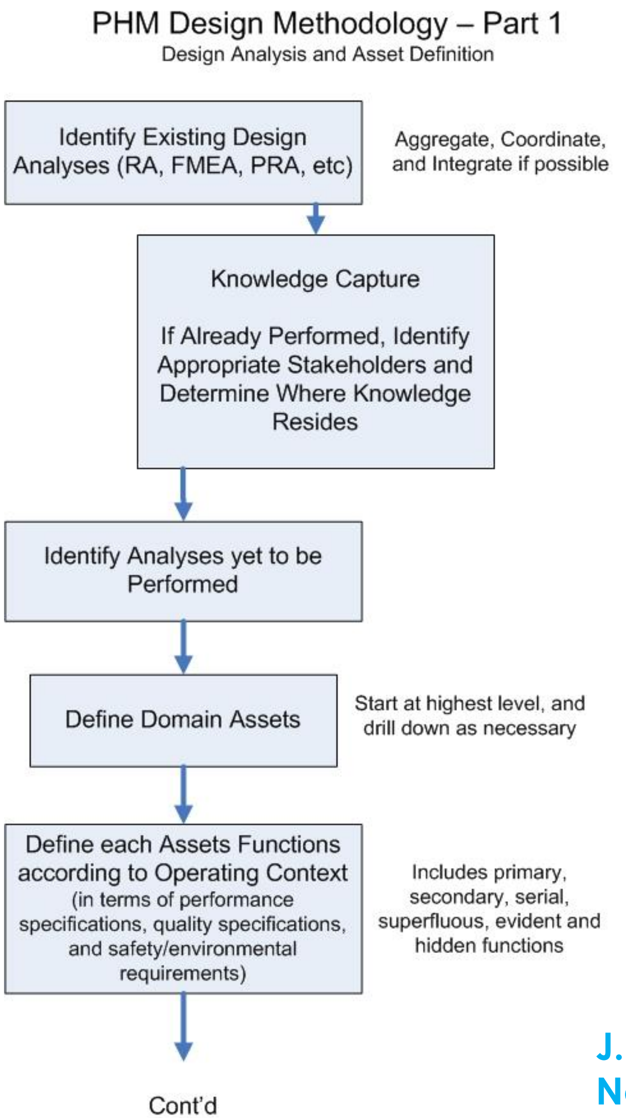
Agile Development Process







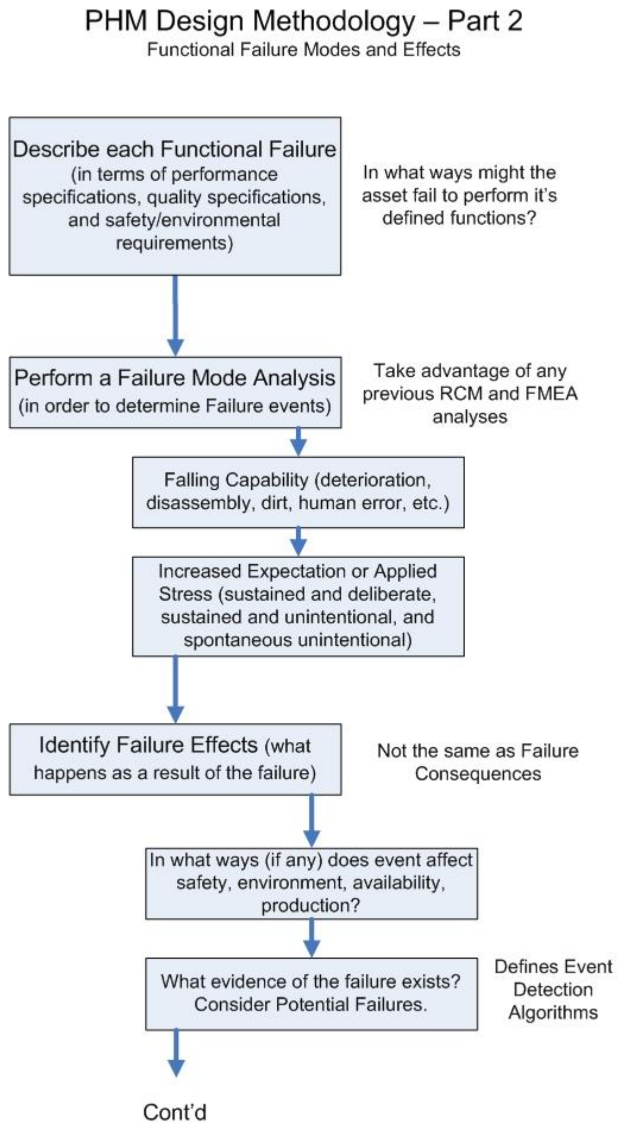
Reliability Centered Maintenance Design



- Reliability Analysis, FMEA, PRA
- Review content and determine if appropriate
- Tools and analyses should support PHM objectives.
- Drawings, specifications, schematics. Model to detail required by critical failure modes

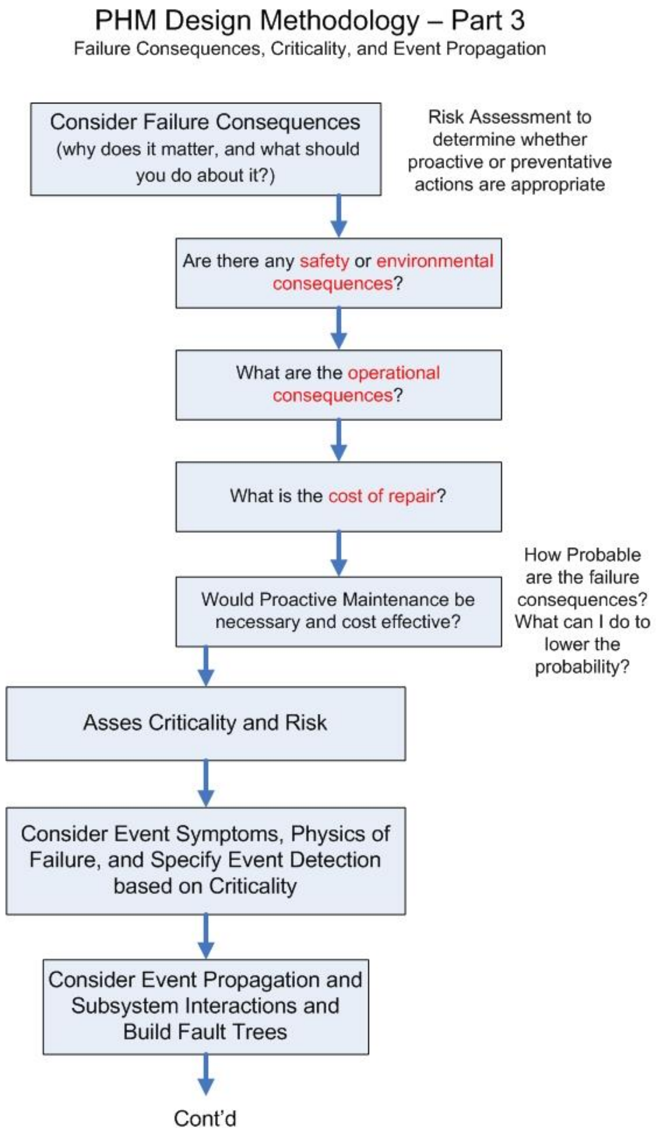
J. Moubray. *Reliability Centered Maintenance, Second Edition.* New York, NY: Industrial Press, 1997.

Reliability Centered Maintenance Design



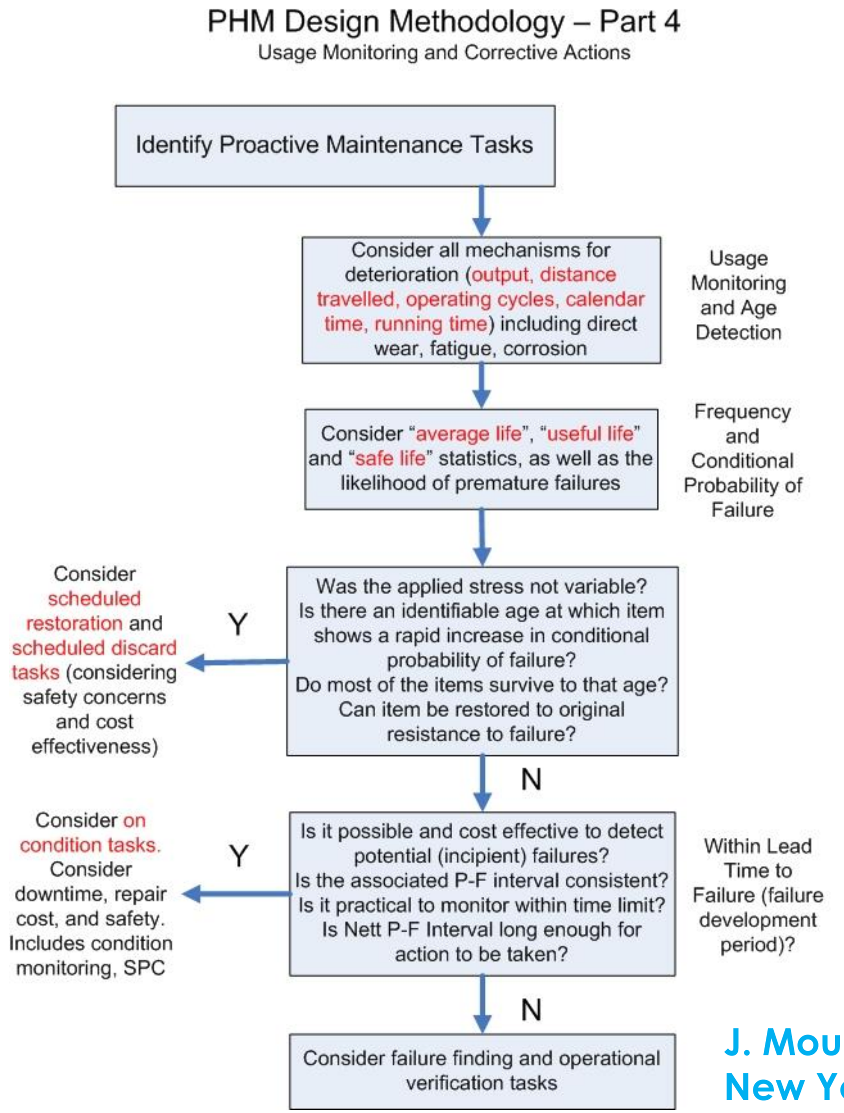
- **Functional Failure descriptions ensure that the PHM system detects what users care about**
- **Analyses may not provide insight into event propagation.**
- **Consider deterioration, increased expectation, and applied stress.**
- **What happens as a result of the failure?**
- **Take the time to consider event propagation. What evidence is available? Consider subsystem interaction**

Reliability Centered Maintenance Design



- **How serious are the effects?**
- **Did something break? Is the system down? Did something spill? Does anyone get hurt?**
- **What could be done to avoid the consequences?**
- **What insight is there for defining event detection logic?**
- **Ready to do fault modeling**

Reliability Centered Maintenance Design

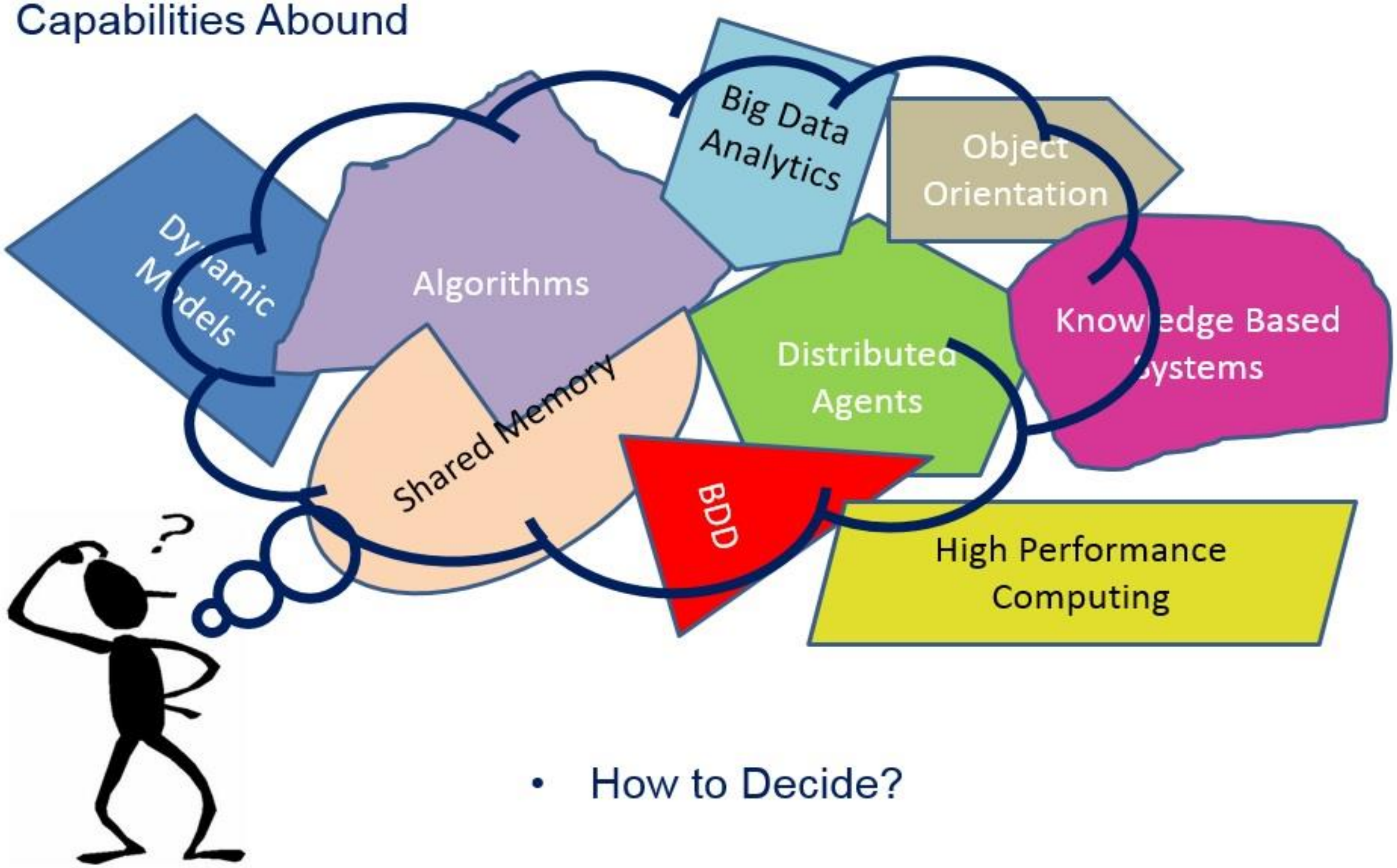


- **Fault detection and isolation vs. Fault prediction**
- **Define the usage monitoring requirements and parameters**
- **Published, estimated, and derived statistical fault likelihoods**
- **Can failure rate be used as specified (and PM scheduled)?**
- **If possible, prognosticate**

J. Moubray. *Reliability Centered Maintenance, Second Edition.* New York, NY: Industrial Press, 1997.

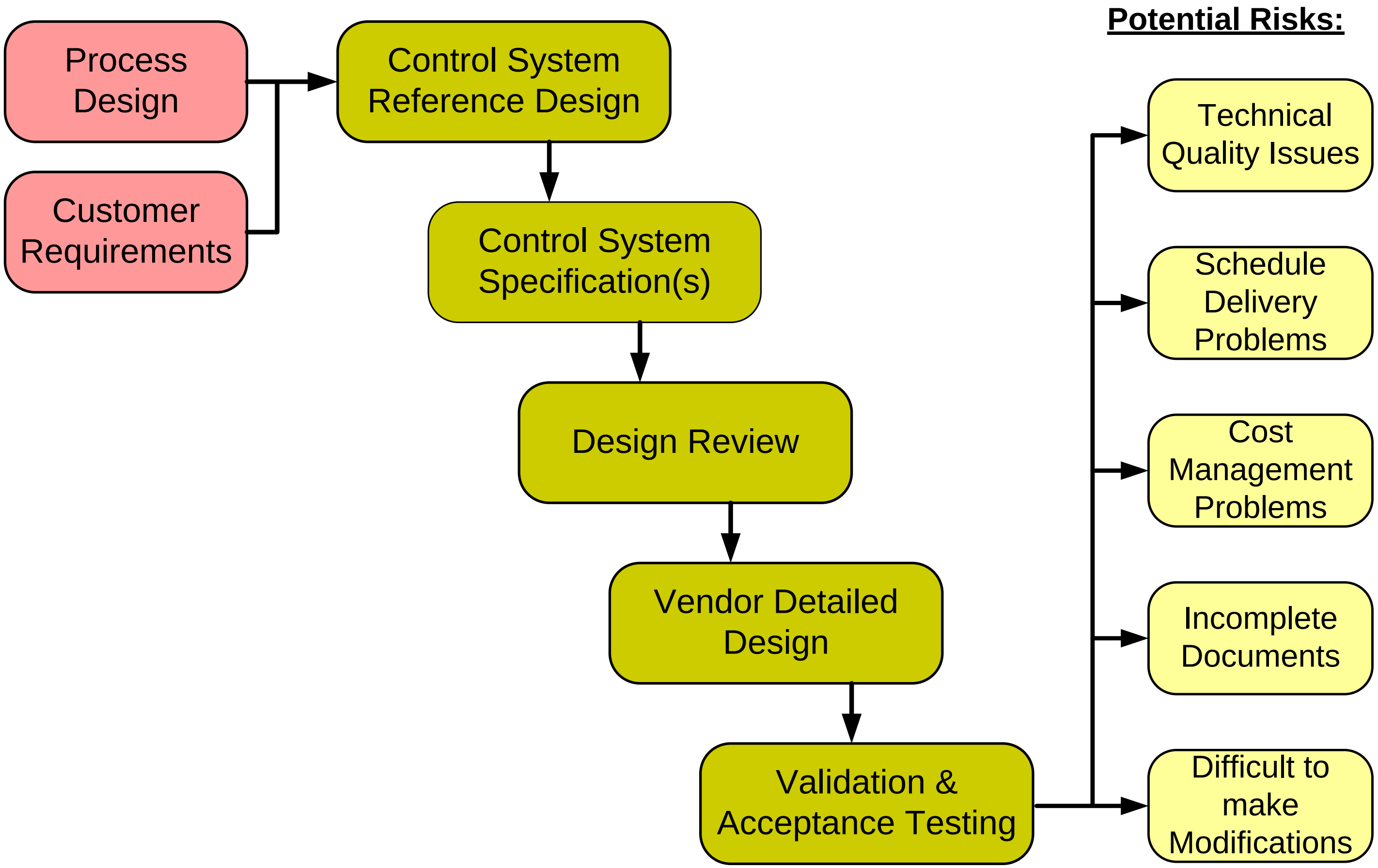
Selecting the Best Architecture

- Capabilities Abound

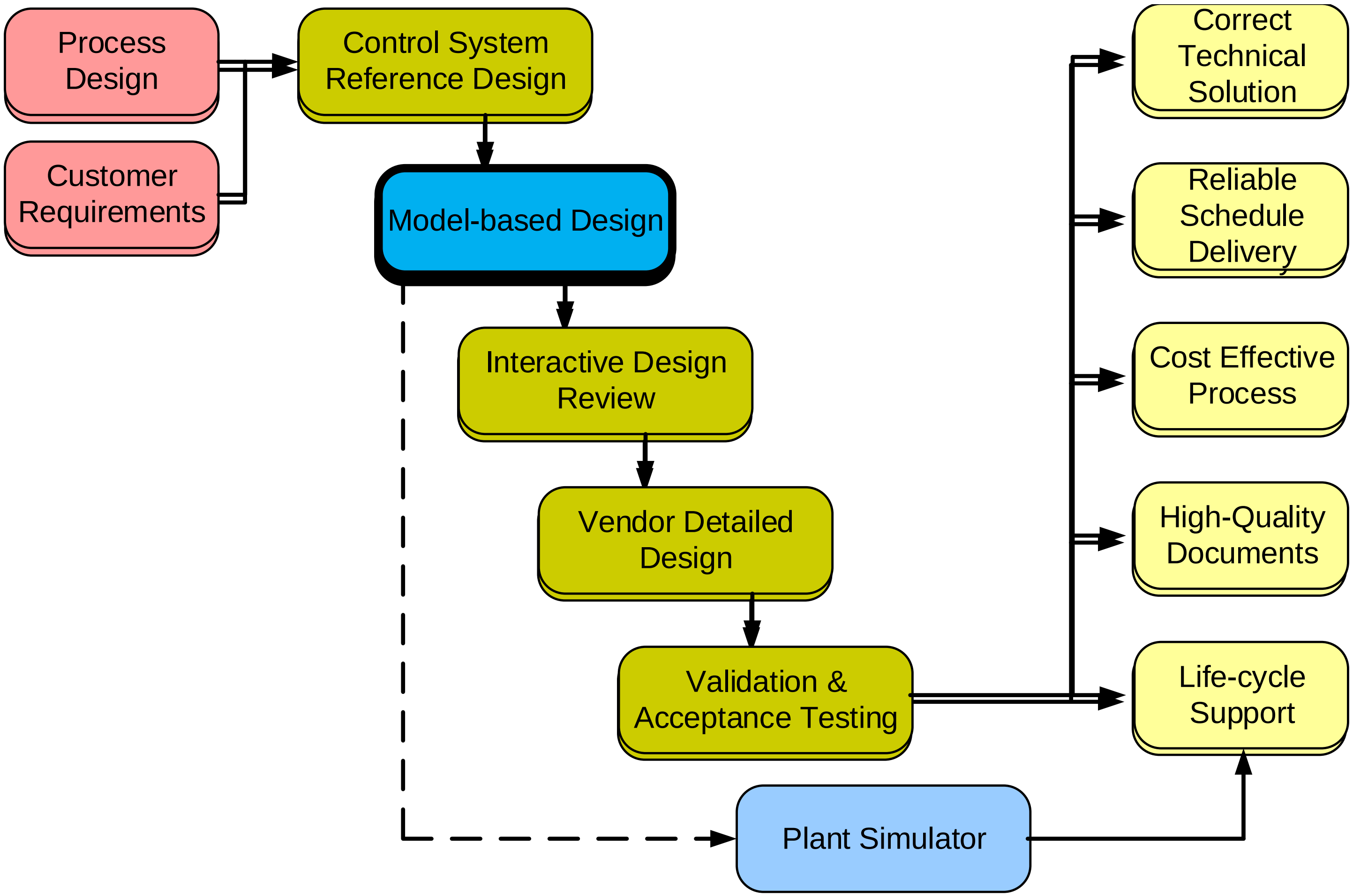


- How to Decide?

Traditional Control System Design

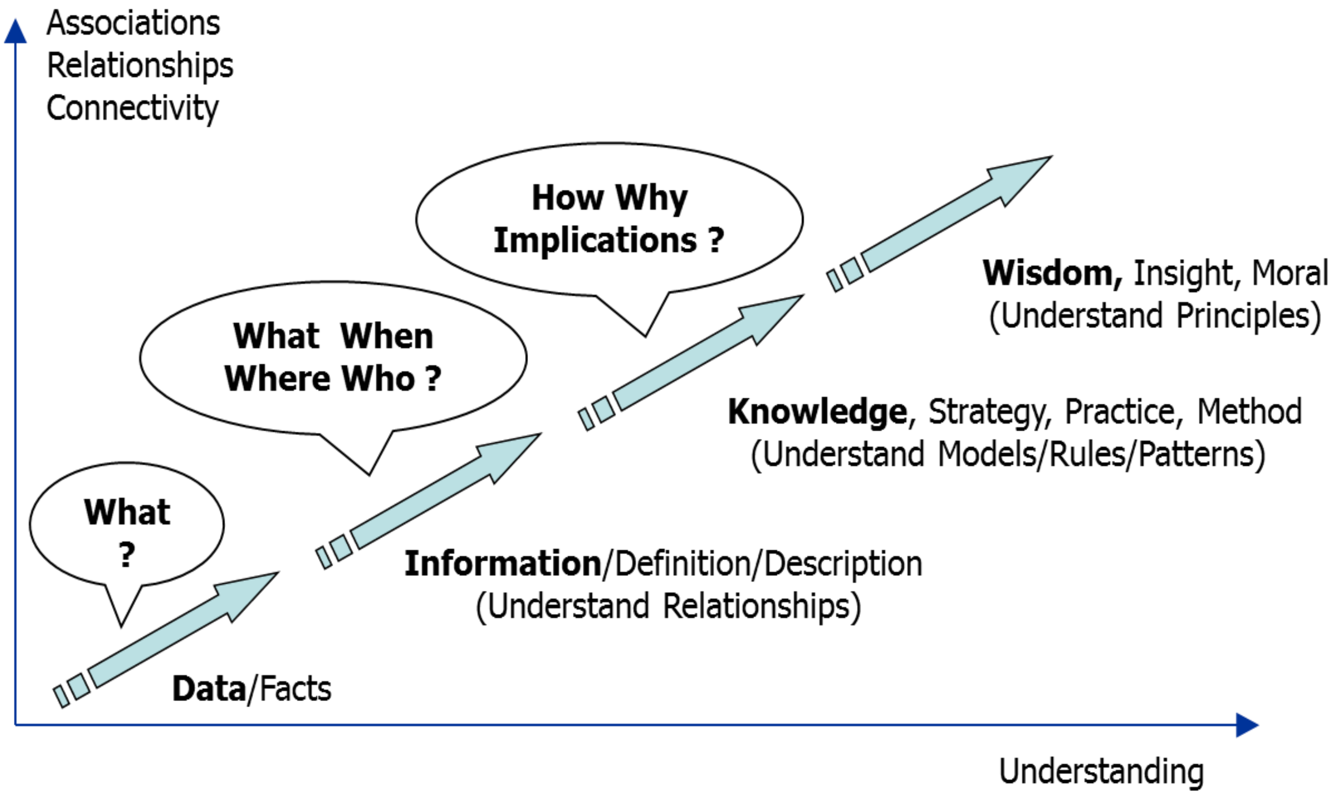


Model Driven Design Improvements

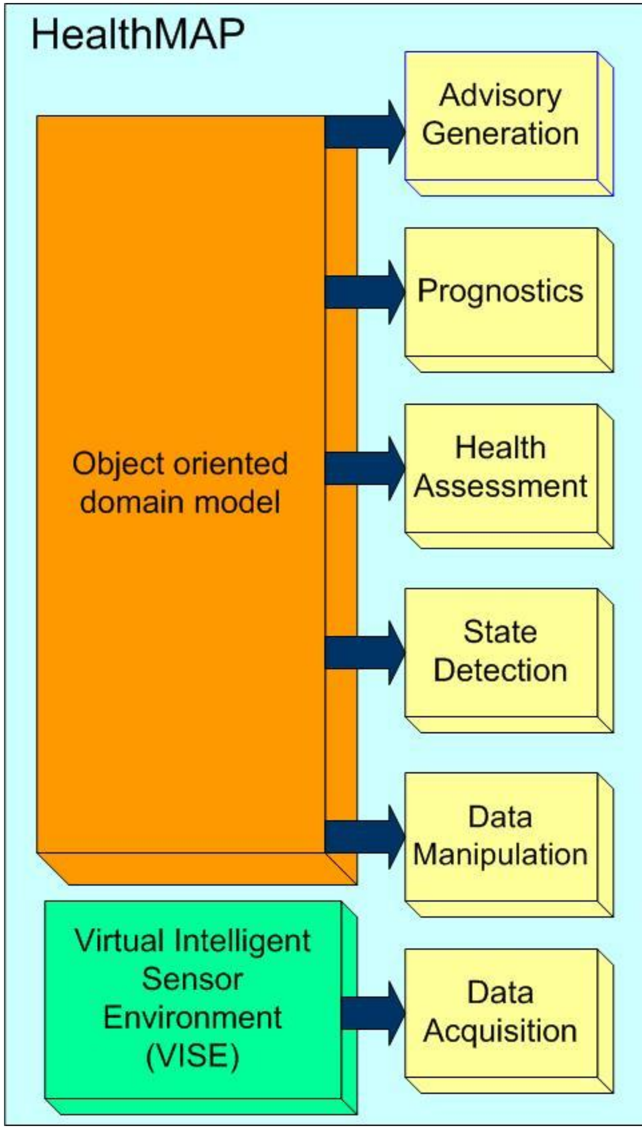


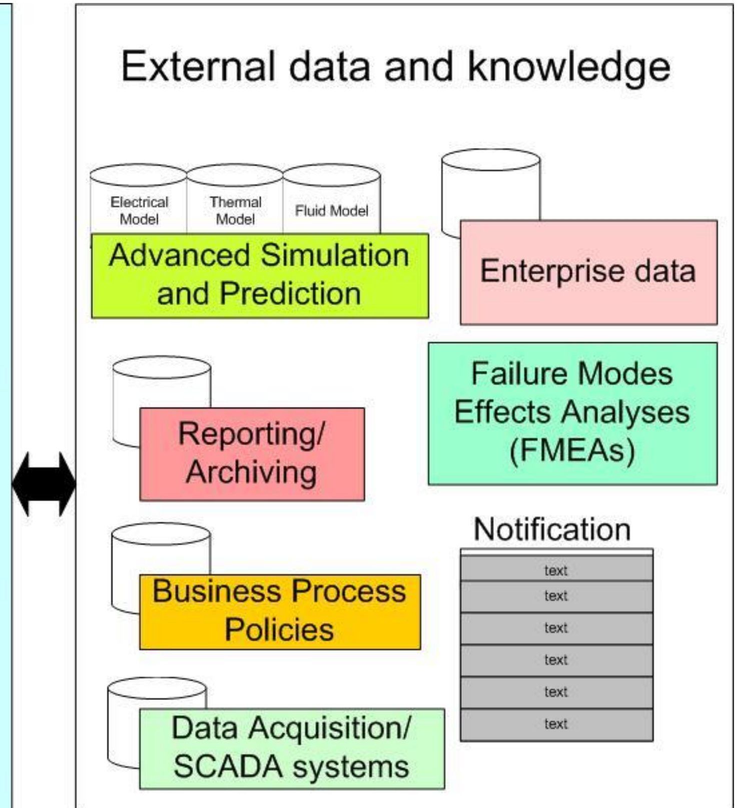
System Model Based Reasoning

Goal: Transform data into information and knowledge based on operational context, leveraging all available wisdom



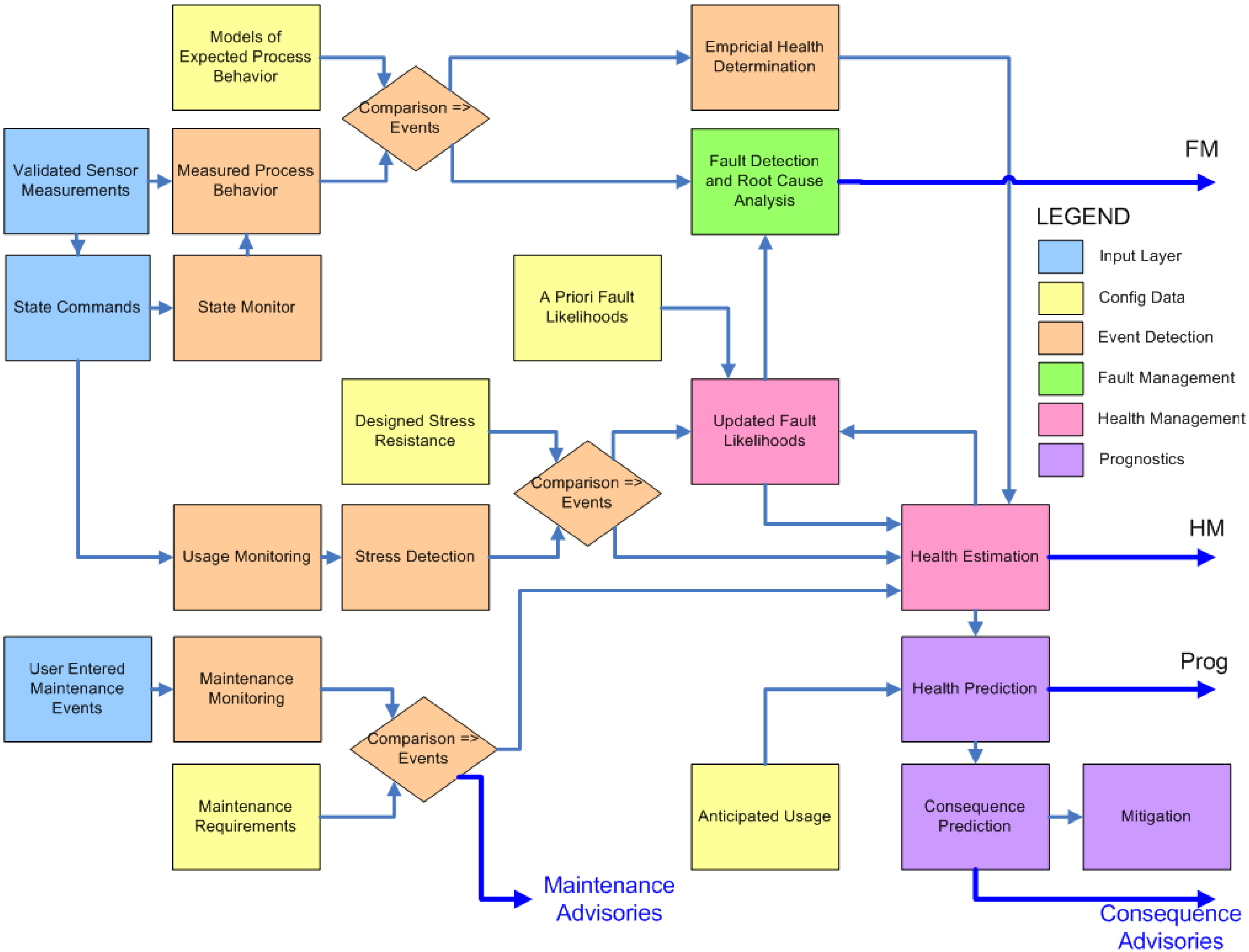
Object Oriented Development Platform





Reference: General Atomics

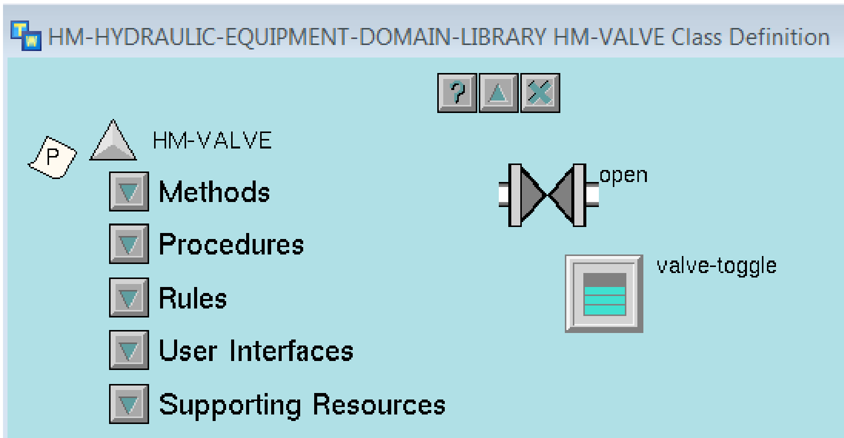
Standards-based Layered Architecture

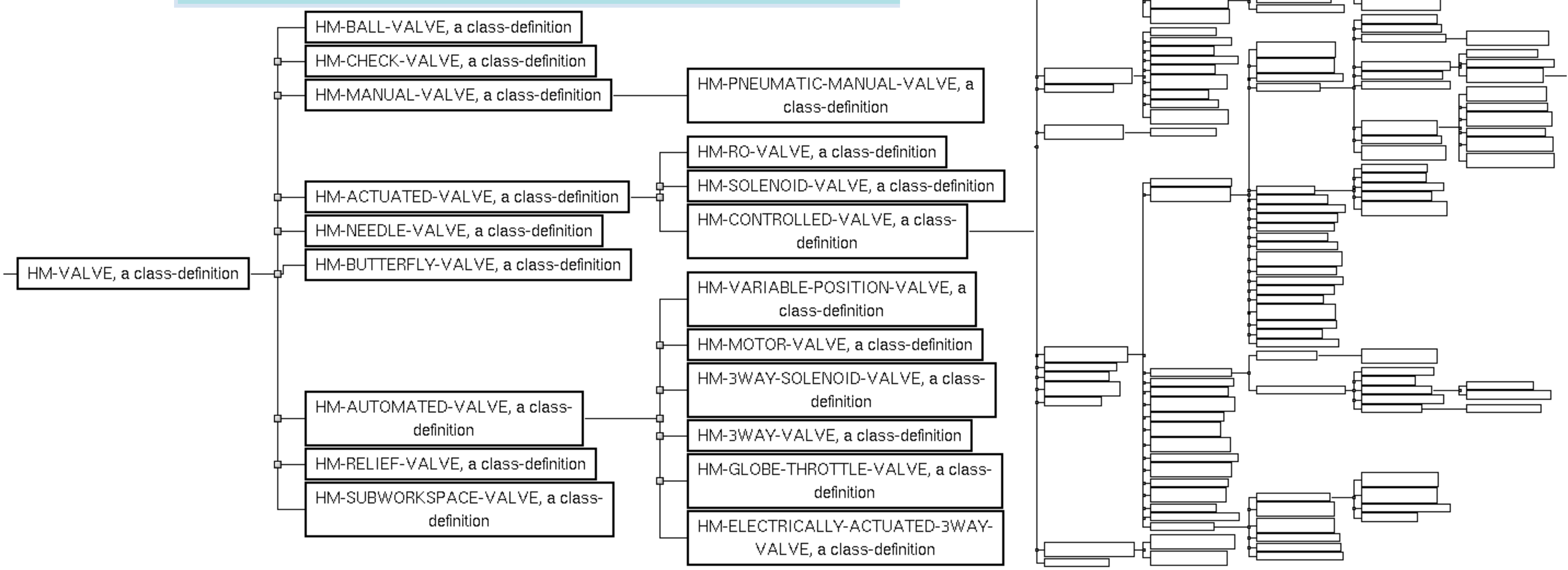


Preferred Platform Characteristics

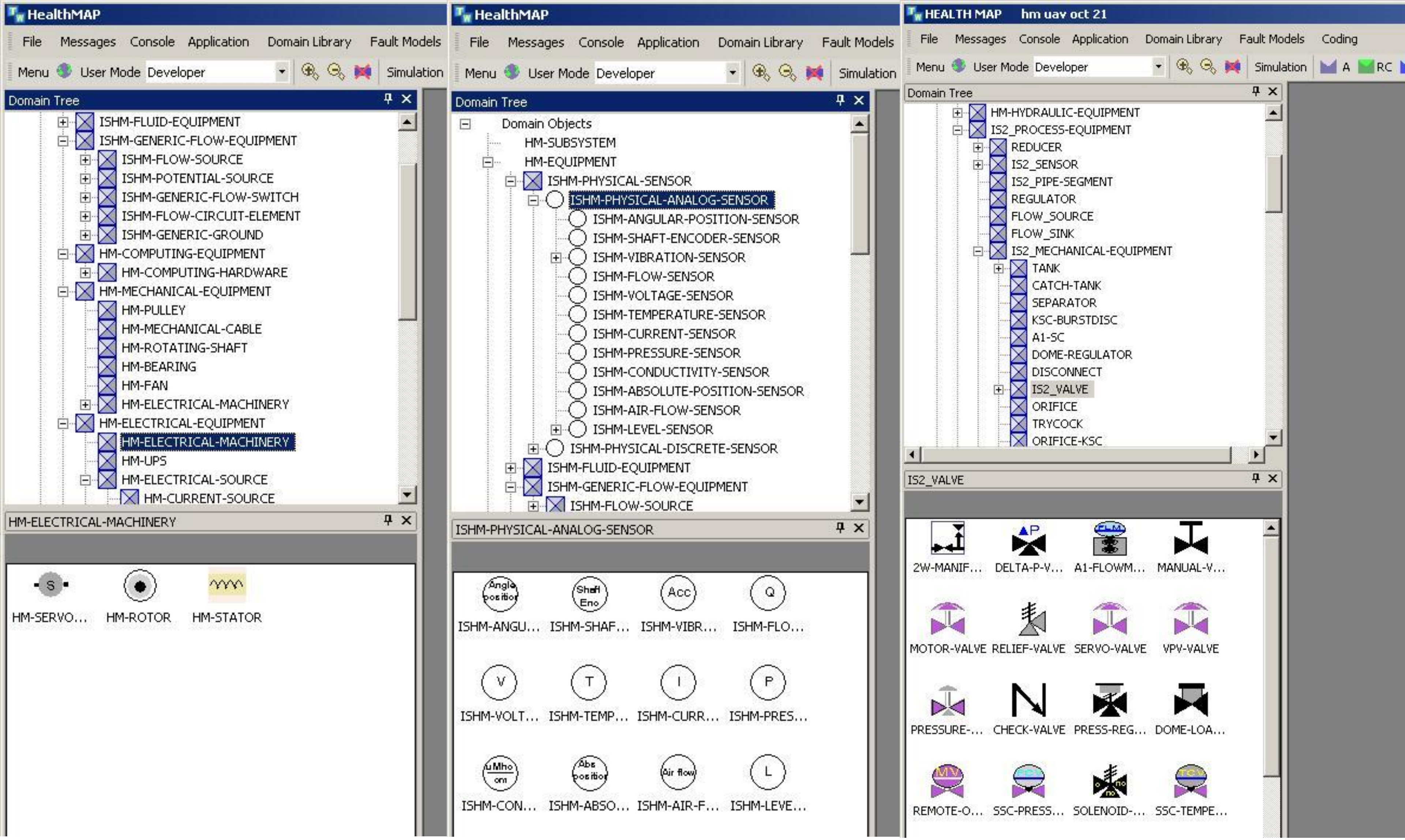
- **Reasoning Execution Engine**
  - Scheduling, simulation, inferencing, trending, state estimation, situational awareness, model-based reasoning, and multi-threaded processing
- **Integrated graphical modeling tools**
  - Domain representation, state transition, fault modeling, neural networks, workflow models, bow-tie diagrams
- **Methodology guided implementation using re-usable libraries**
- **OSA application supporting standards-based interfacing**
  - Transducers, DACs, PLCs, DCSs, SCADA, data aggregation platforms, 3rd party management tools, dynamic modeling and simulation platforms, enterprise data, Plant Historian, end user notification

Complex Generic Objects with Supporting Methods

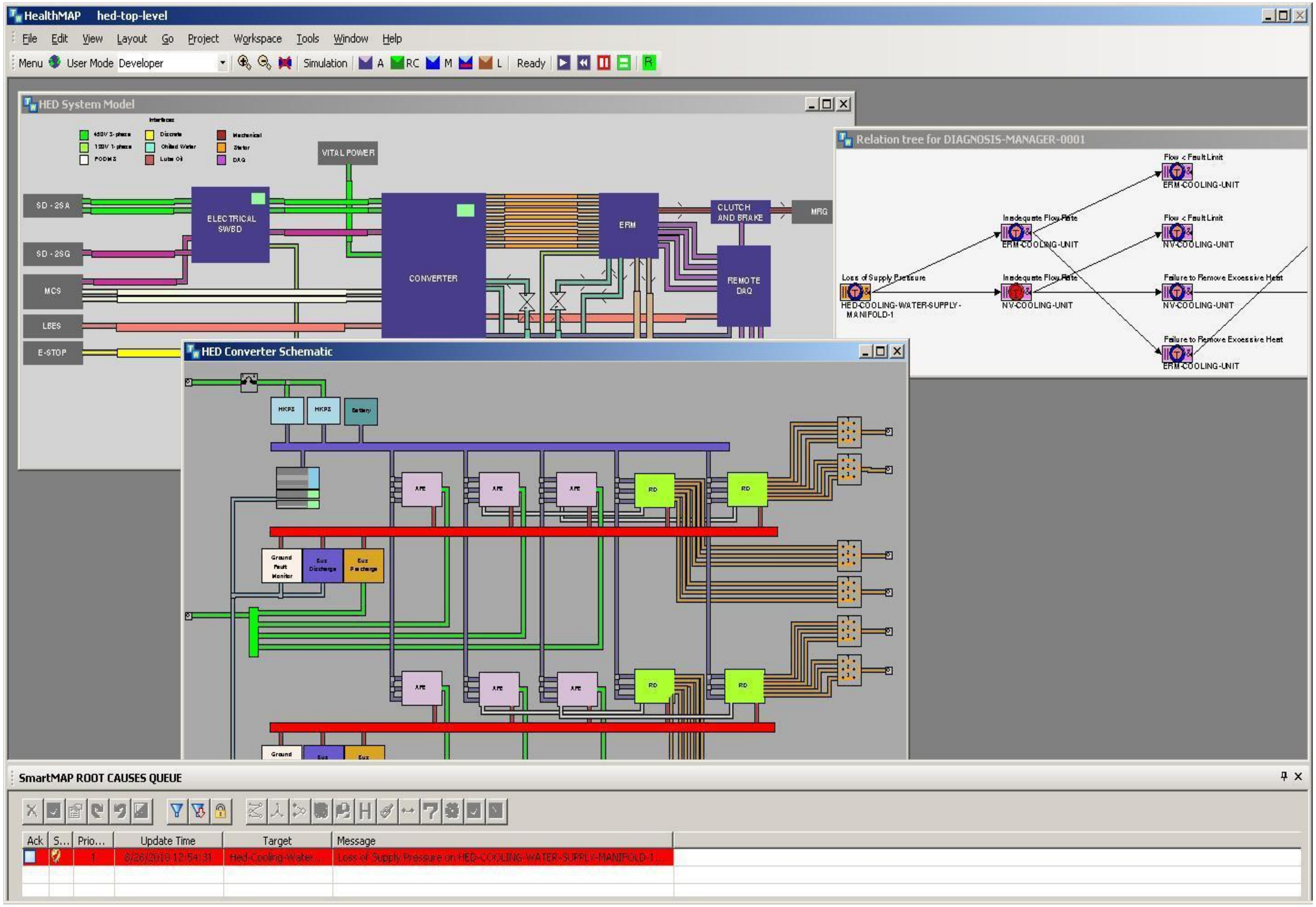




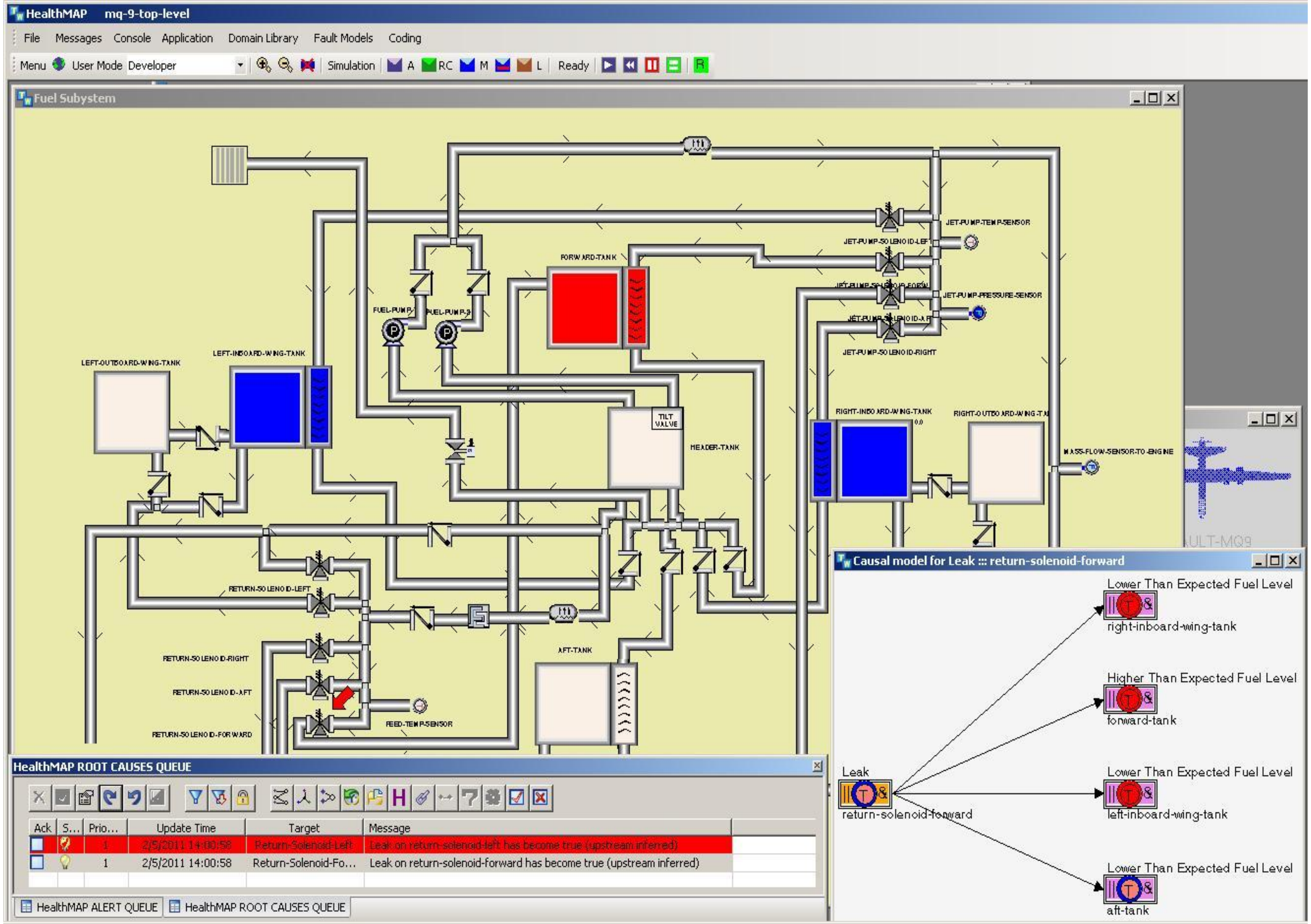
Extensible Model Libraries w/ Palettes



Domain Representations

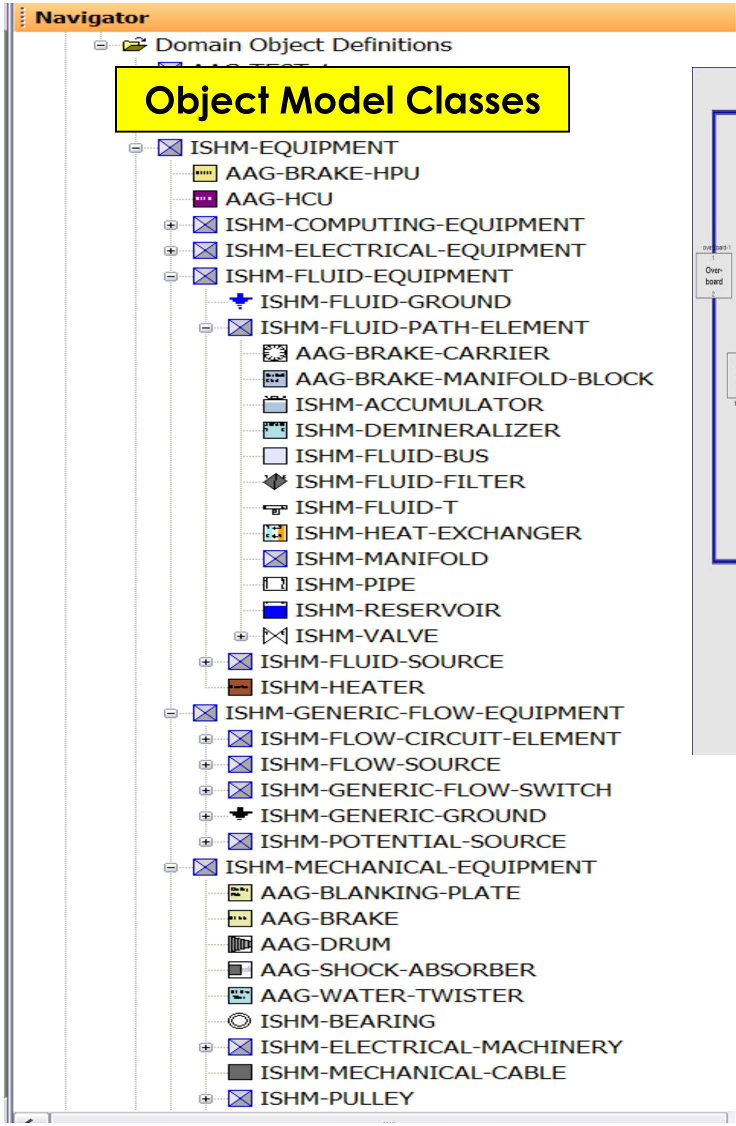


Domain Representations

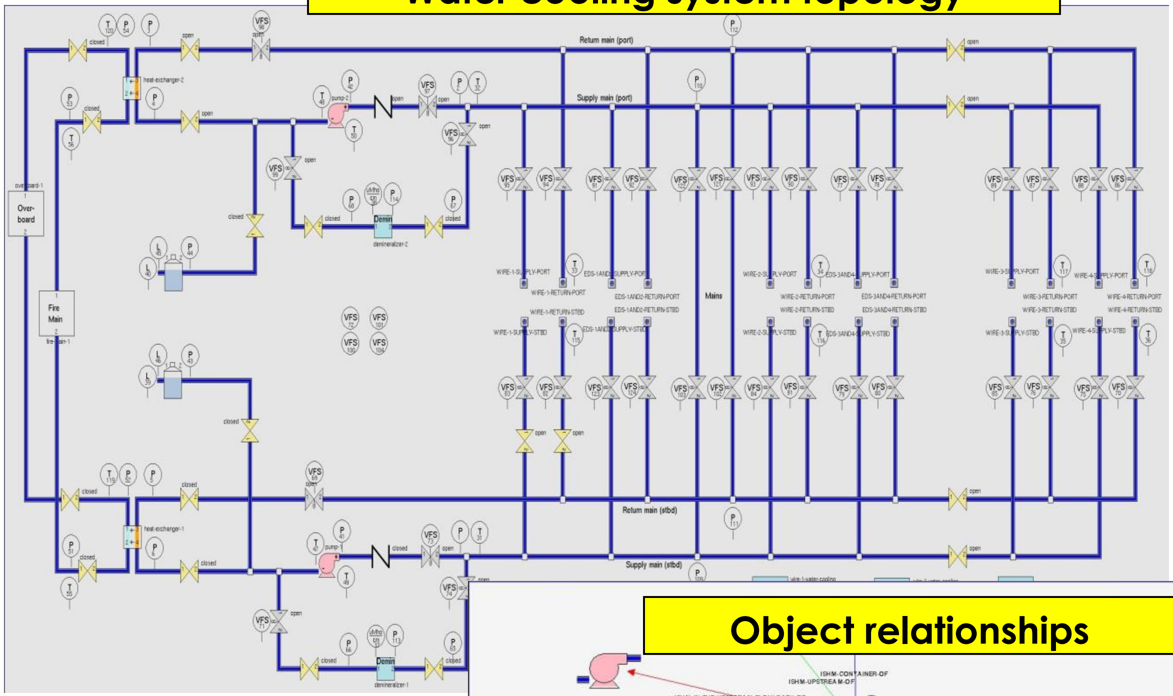


Relational Modeling Support

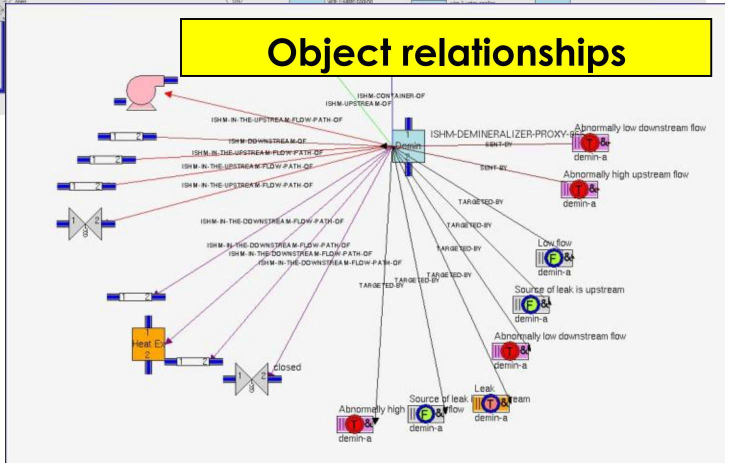
Object Model Classes



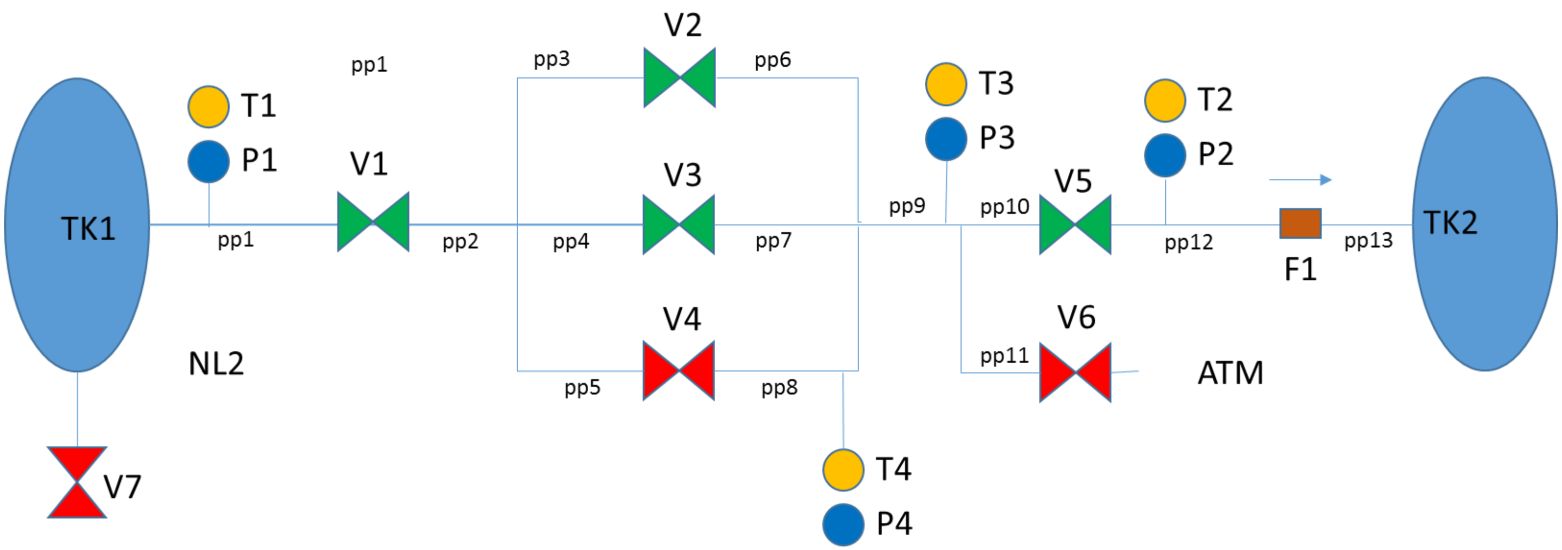
Water cooling system topology



Object relationships



Concepts and Models



Flow Subsystem as a Concept

Flow Subsystem 1: Members (TK1, pp1, T1, P1, pp2, pp3, V2, pp6, pp9, T3, P3, V5, T2, P2, F1, TK2), Source: TK1, Sink: TK2.

Flow Subsystem 1: Members (TK1, pp1, T1, P1, pp2, pp4, V3, pp7, pp9, T3, P3, V5, T2, P2, F1, TK2), Source: TK1, Sink: TK2.

Note: AO-MDS incorporates the concept of Flow Subsystem and dynamically determines Flow Subsystems for any application and its current configuration.

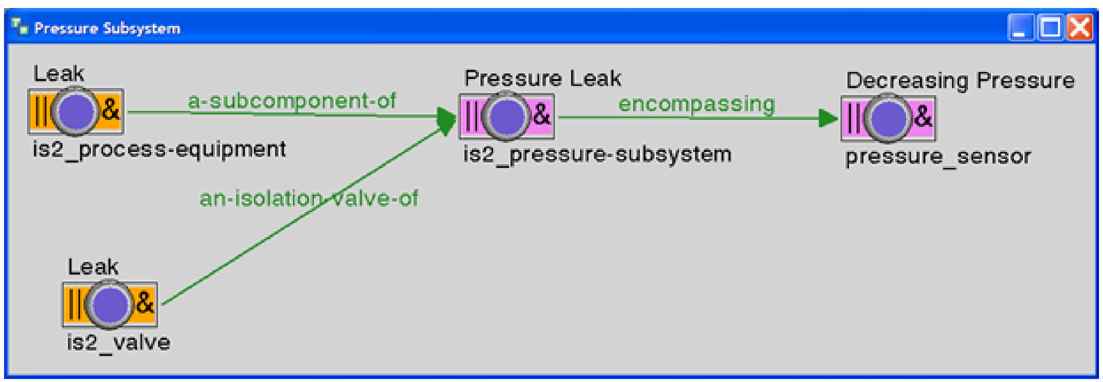
In Contrast with a data/information driven approach:

Flow subsystem selected from a pre-defined list that considers all possible combinations of valve configurations for all schematics

- generally hundreds or thousands of valves are involved, becoming a complex combinatorial problem.
- Any changes in the system (e.g. adding a valve) will require extensive work to update the combinatorial list.
- Any new system will require its own combinatorial list.

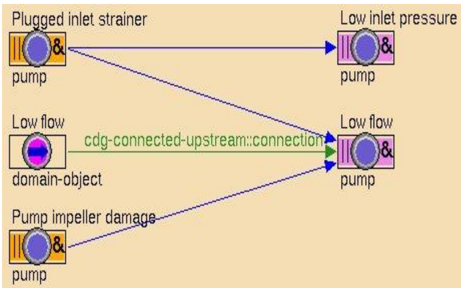
Failures Modes and Effects Analysis (FMEA) Modeling based on MIL-STD-1629A(2)

ID #	Item-Functional Identification	Function	Failure Modes and Causes	Mission Phase-Operational Mode	Failure Effects			Failure Detection Method
					Local End Effects Effects	Next Higher Level		
	Process Equipment	Fluid feed subsystem	Leak	Sealed subsystem maintaining pressure		Pressure leak	Decreasing pressure measurement	Identify sealed subsystem, and check pressure sensors for decreasing pressure.

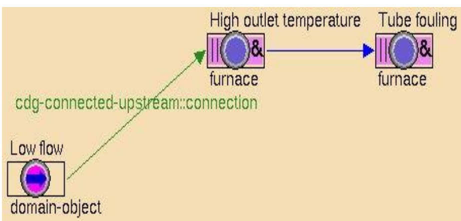


Fault Detection, Diagnosis, and Prognosis

Generic Fault Models	+	Domain Map	=	Specific Fault Model



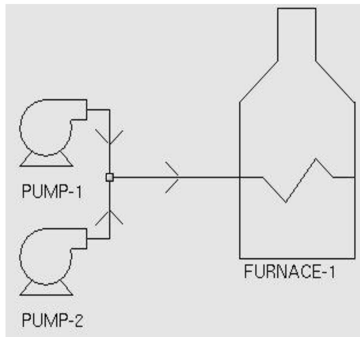
Pump generic fault model



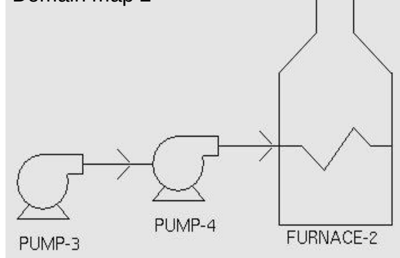
Furnace generic fault model

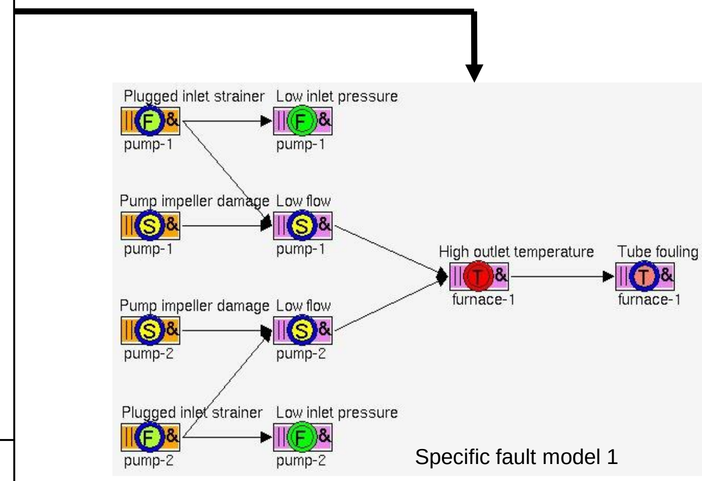
Test and repair actions

Domain map 1

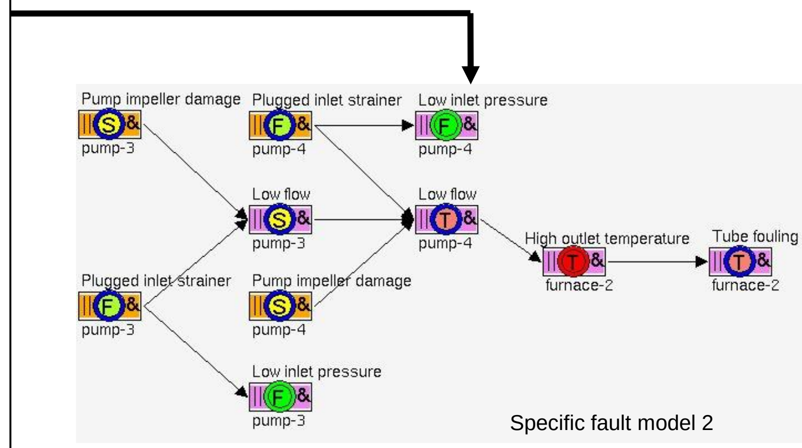


Domain map 2





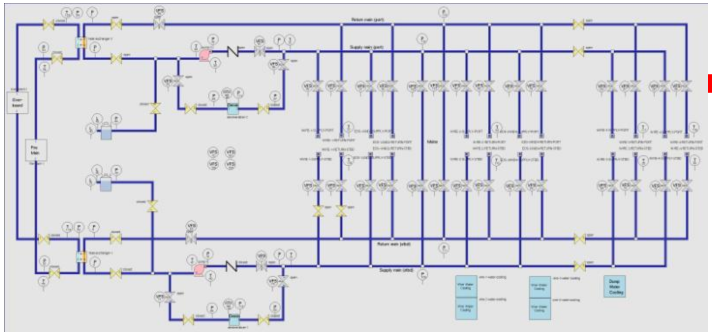
Specific fault model 1

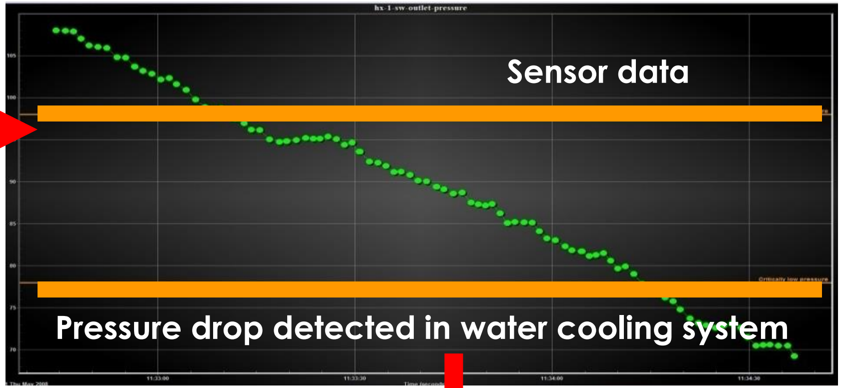


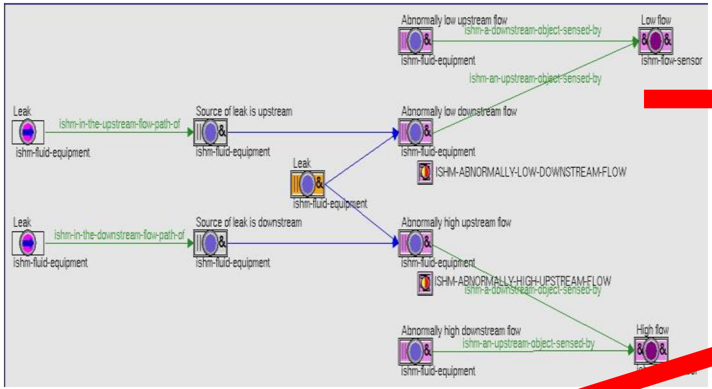
Specific fault model 2

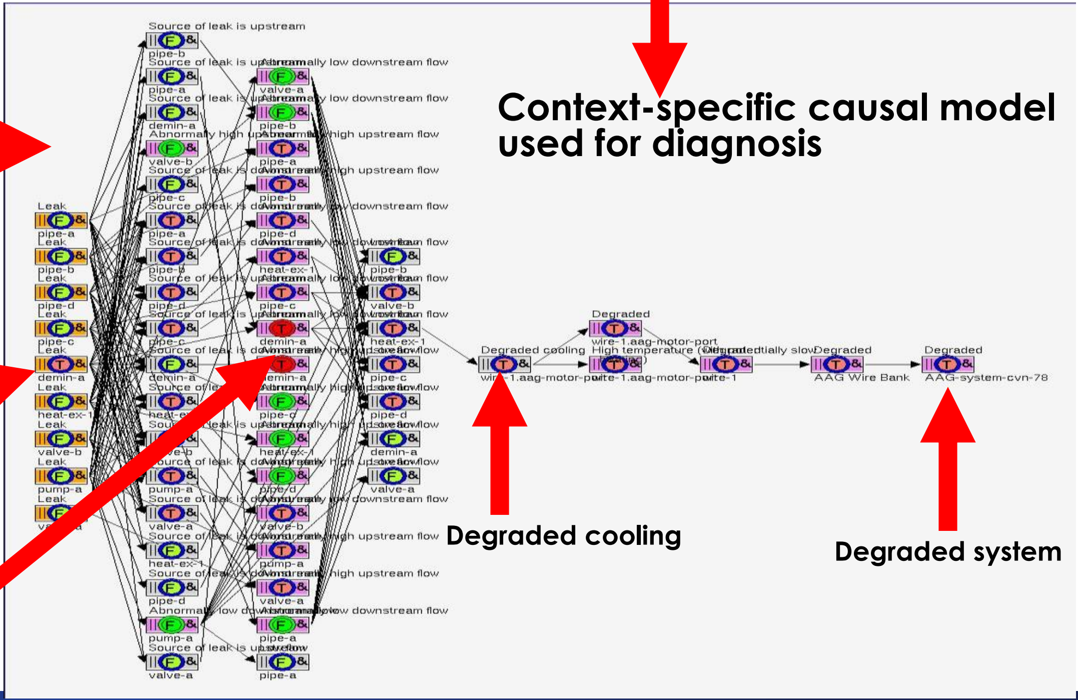
Model based reasoning: event detection and diagnosis

Domain model used to predict expected values









Context-specific causal model used for diagnosis

Valve leak

Low pressure downstream of leak

Degraded cooling

Degraded system

Debugging Specific Fault Models