

Experiences in Applying SysML to Develop Interoperable Torpedo Modeling and Simulation Components

Presented to:

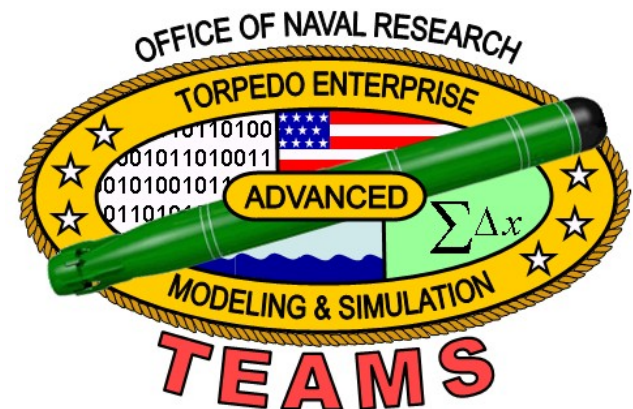
NDIA

10th Annual Systems Engineering
Conference
San Diego, CA

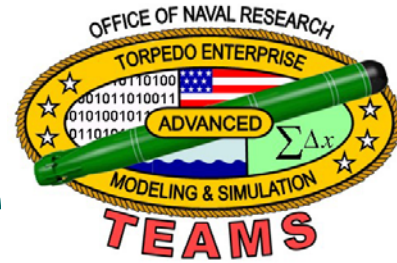
Presented by:

Thomas Haley
Naval Undersea Warfare Center
Division Newport
Newport, RI

24 October 2007

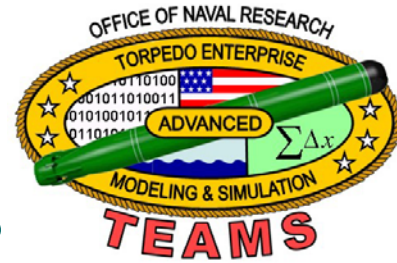


Outline



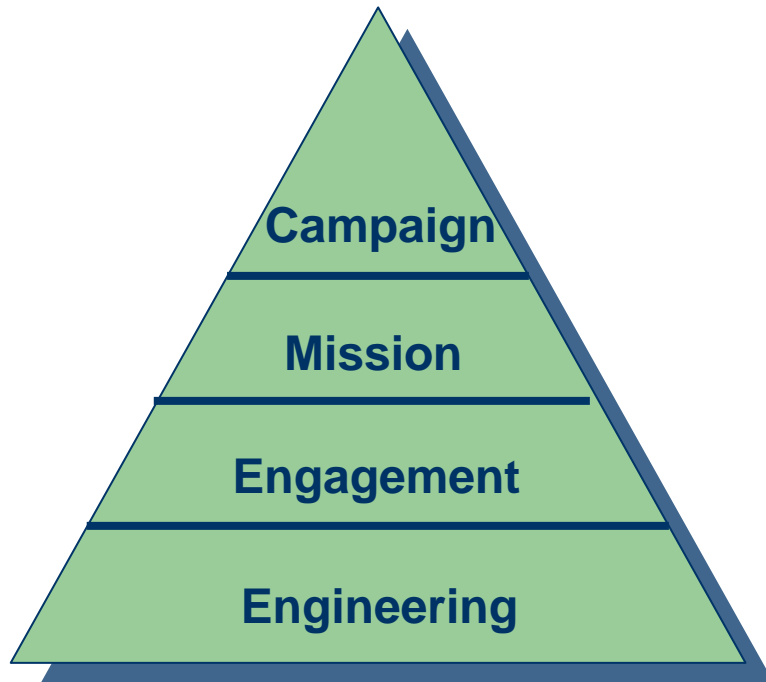
- **SysML Case Study Motivation**
- **TEAMS Project Background**
- **SysML Proof of Concept**
- **Lessons Learned**
- **TEAMS Perspective: SysML Pros and Cons**
- **Acknowledgements**

Motivation: Feasibility of Open Standards



- **Funded by Office of Secretary of Defense, Systems and Software Engineering**
- **Determine if open standards can be used to describe:**
 - **System of systems (SoS) architectures based on computer models**
 - **System components as elements of composable distributed simulations**
- **Determine whether SysML models can be used in conjunction with performance simulation models**

Background: TEAMS Simulation Scope

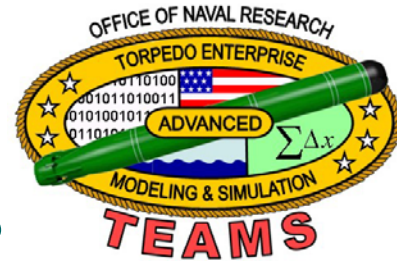


**TEAMS Emphasis:
“Launch-to-Hit”
Analysis**

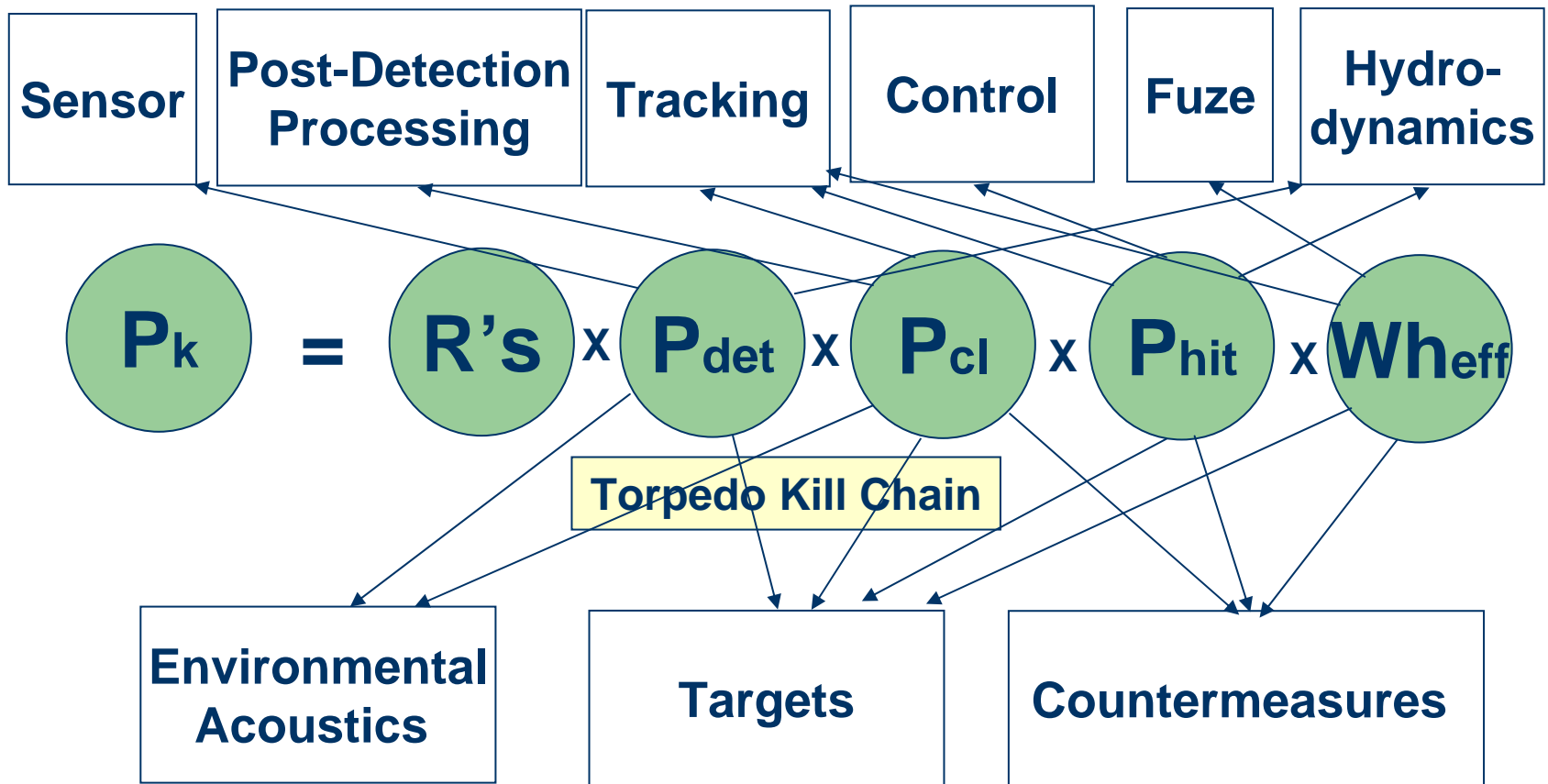
Military M&S Resolution Levels

TEAMS: Torpedo Enterprise Advanced Modeling & Simulation

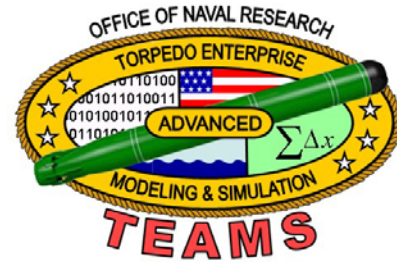
Background: High-Level M&S Requirements



Torpedo M&S Components

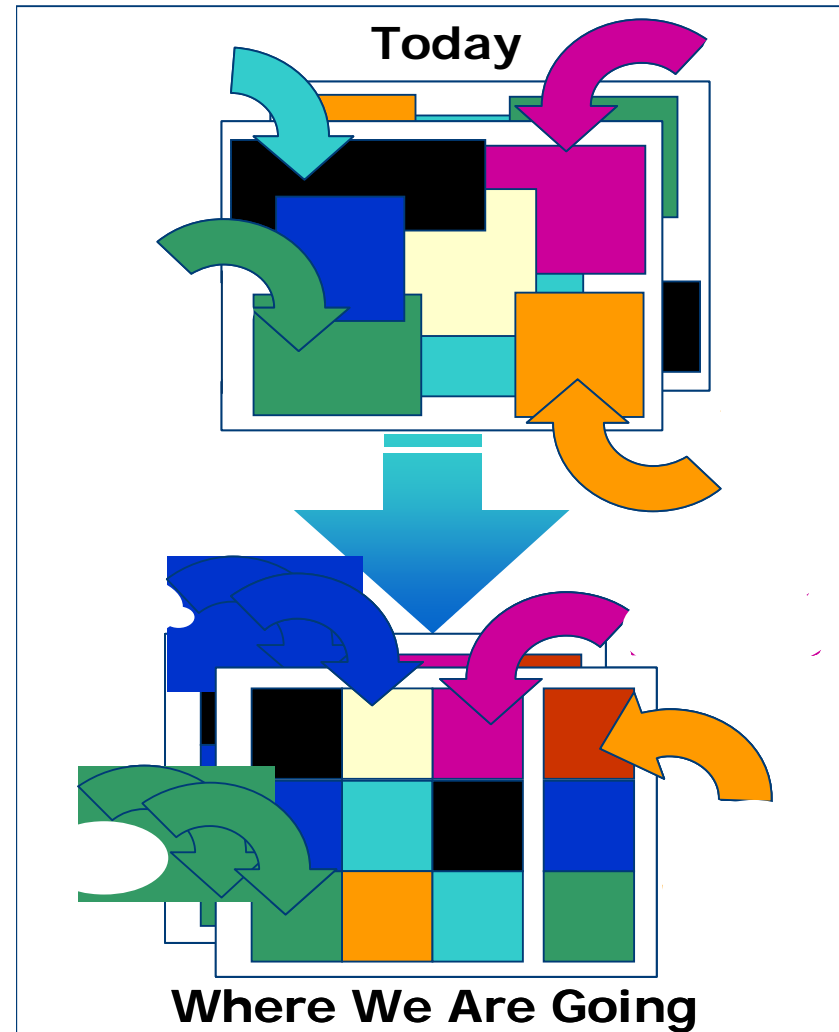


Other "Stimulus" M&S Components



TEAMS Background

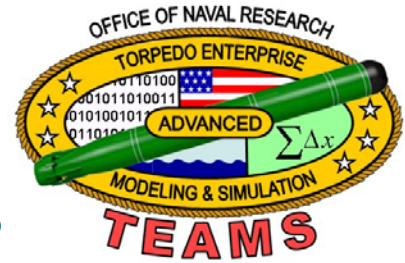
- **Problem: Modeling & Simulation Business “Model” Obsolete**
 - Monolithic
 - Stove pipes
 - Single developers
 - No communication
- **Solution: Foster Collaborative M&S Development Environment**
 - Standardize M&S architecture framework and component models
 - Reduce the technology development timeline
 - Increase model content, implementation efficiency and reuse
 - Reduce cost





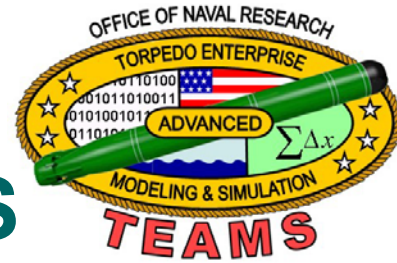
Overall TEAMS Goals

- **Modeling and Simulation Community Collaboration**
- **Standardized architecture framework**
 - **Conceptual reference model**
 - **Model-based requirements specifications**
- **Standardized reference model interfaces**
 - **Interchangeable & composable components**
 - **Extendable to other applications (e.g., XML schema)**
 - **Semantically described (e.g., OWL ontology)**
- **Document standards and requirements**
- **Cost effective process to achieve interoperability and composability**
- **Business model for future cross-organization M&S funded efforts**



TEAMS Core Requirements

1. Standard Interfaces
2. Platform Independence
3. Open Standards
4. Model Realizable Systems
5. Extensible Interfaces
6. Evolving Standards
7. Loosely Coupled Interfaces
8. Tiers of Interfaces
9. Support Different Levels of Detail
10. Standard Implementation Strategies



Organizations Looking to TEAMS

THE *Open* GROUP

International organization, developers of TOGAF architectural framework

- Wants TEAMS as test case for TOGAF 8.1.1 and 9.0
- Interest in using TEAMS to test synergy between DoDAF and TOGAF frameworks
- Wants TEAMS for its process to incorporate Ontologies (relationships of components)



International organization, developers of several business communications standards

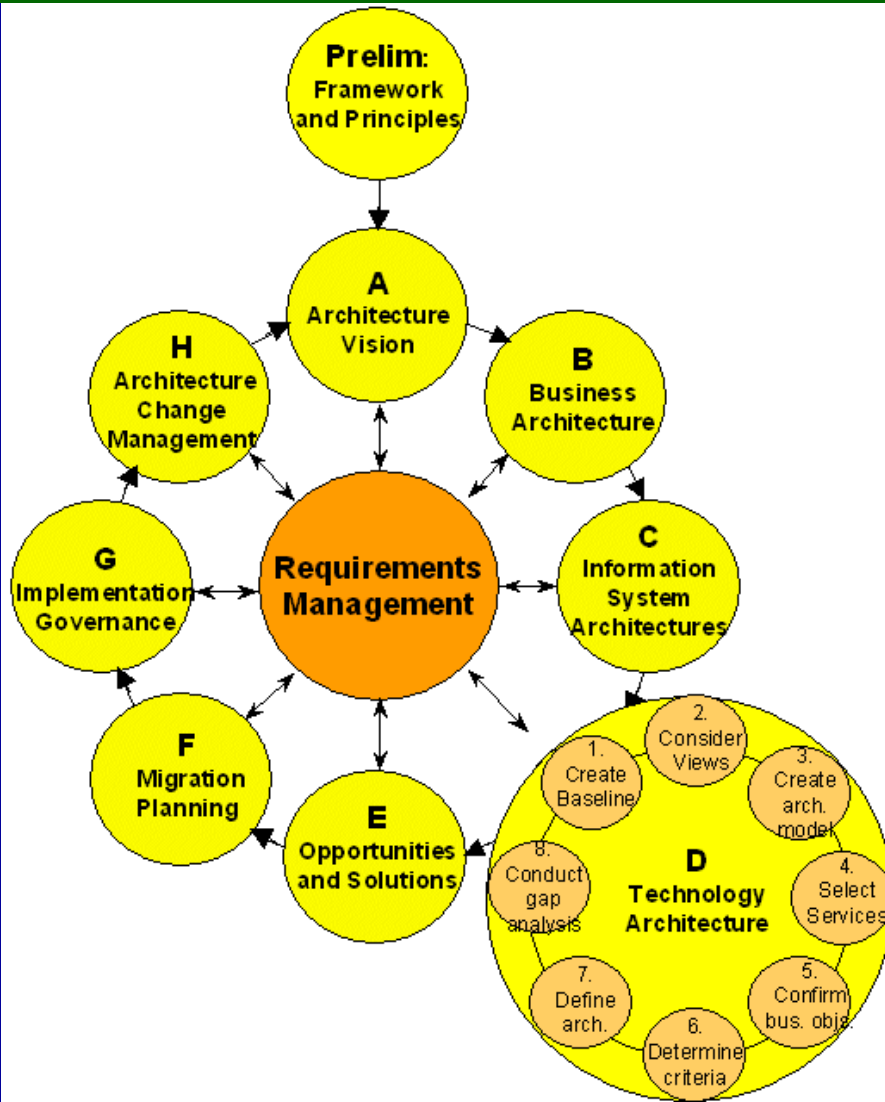
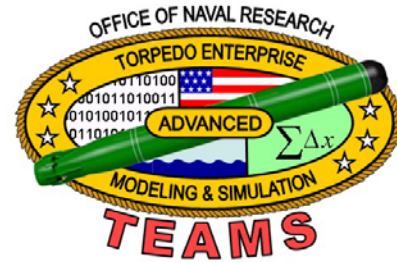
- Used TEAMS as test case for their TOGAF/ Model Driven Architecture (MDA) under the TOGAF/MDA Synergy Project



The Open Systems Joint Task Force of the Office of Secretary of Defense (OSD)

- Wants to convert TEAMS UML artifacts to the newly approved SysML standard to demonstrate utility of the new standard

High-Level Process: TOGAF ADM



The Open Group:
IT Consortium
Offers Consortia Services

TOGAF:
The Open Group
Architecture Framework

ADM:
Architecture
Development Method

“OMG™ is [a] ... not-for-profit computer industry consortium ... developing enterprise integration standards for a wide range of technologies [... / ...] industries ... enabl[ing] powerful visual design, execution and maintenance of software and other processes...”

- CORBA – Common Object Request Broker
- UML – Unified Modeling Language
- SysML – Systems Engineering Modeling Language
- Numerous others in diverse industries (e.g., business)
- Developer of Model Driven Architecture (MDA) method

OMG has a model-based emphasis in developing standards

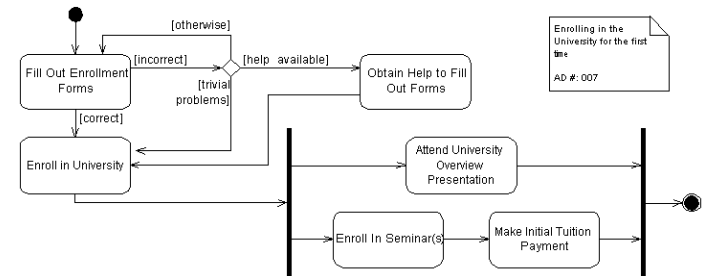
UML

UML Consists of 13 Diagrams

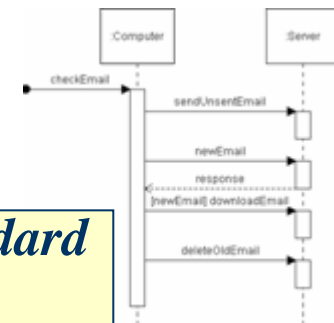
Structure: E.g., Class Diagram



Behavior: E.g., Activity Diagram

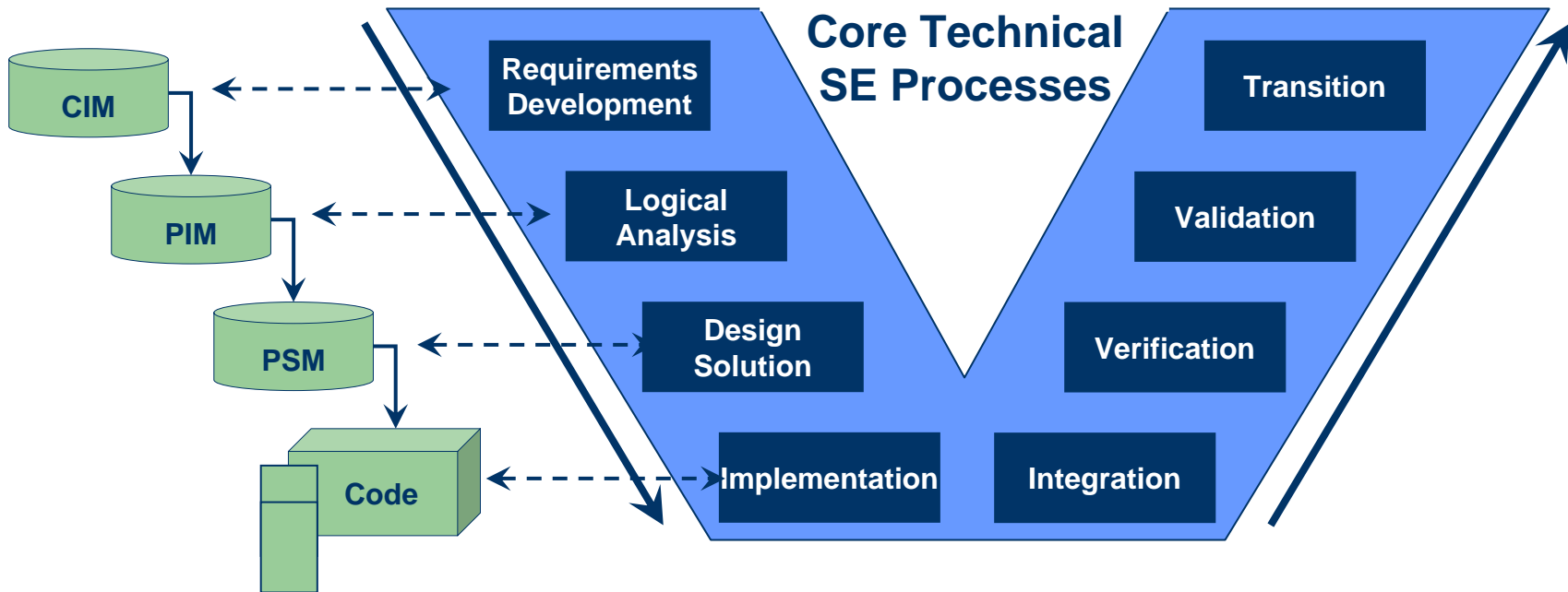


Interaction: E.g., Sequence Diagram



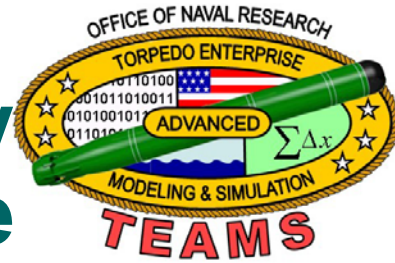
*OMG models are MOF-Based - Meta-Object Facility Standard
 Think "TurboTax"*

Using MDA in SE Context



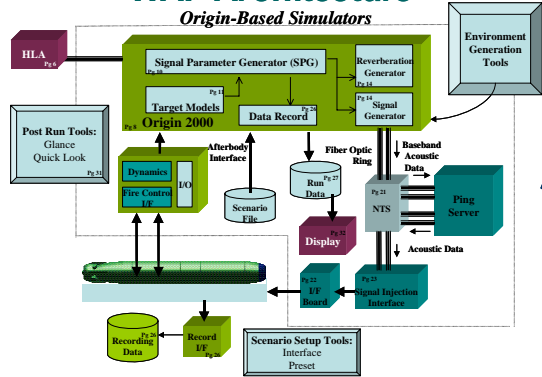
The implementation (code) for technology selected by the developer

Baseline Technology Architecture



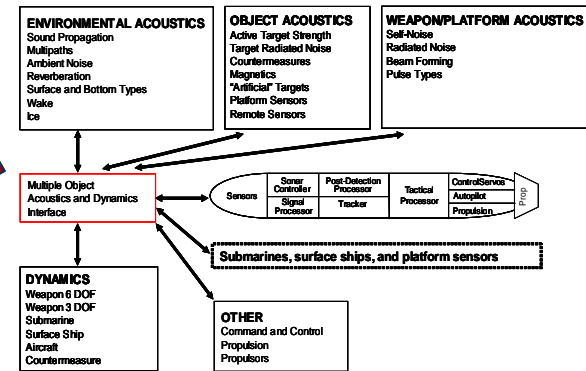
Weapons Analysis Facility (WAF)

WAF Architecture



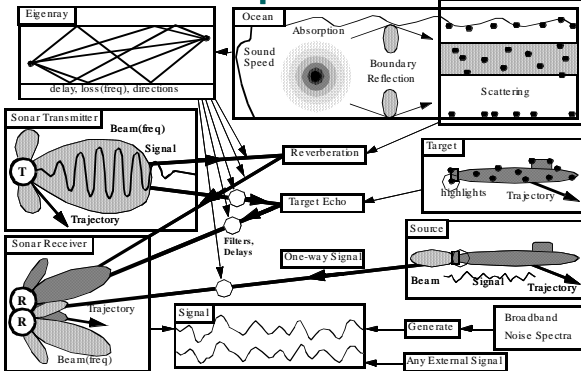
Technology Requirements Model (TRM)

TRM's CARLEE

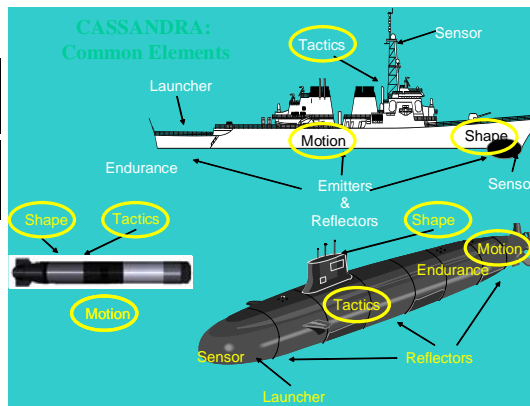


Conceptual Reference Model

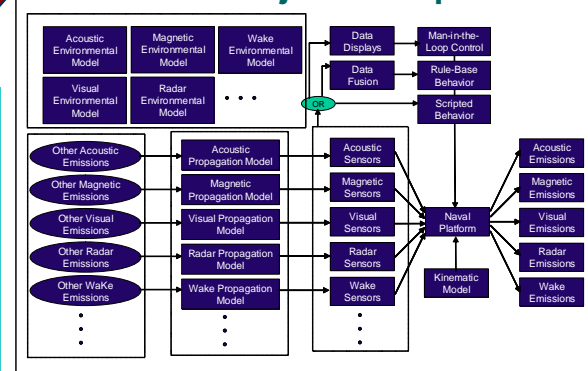
SST Component Models



CASSANDRA Common Elements

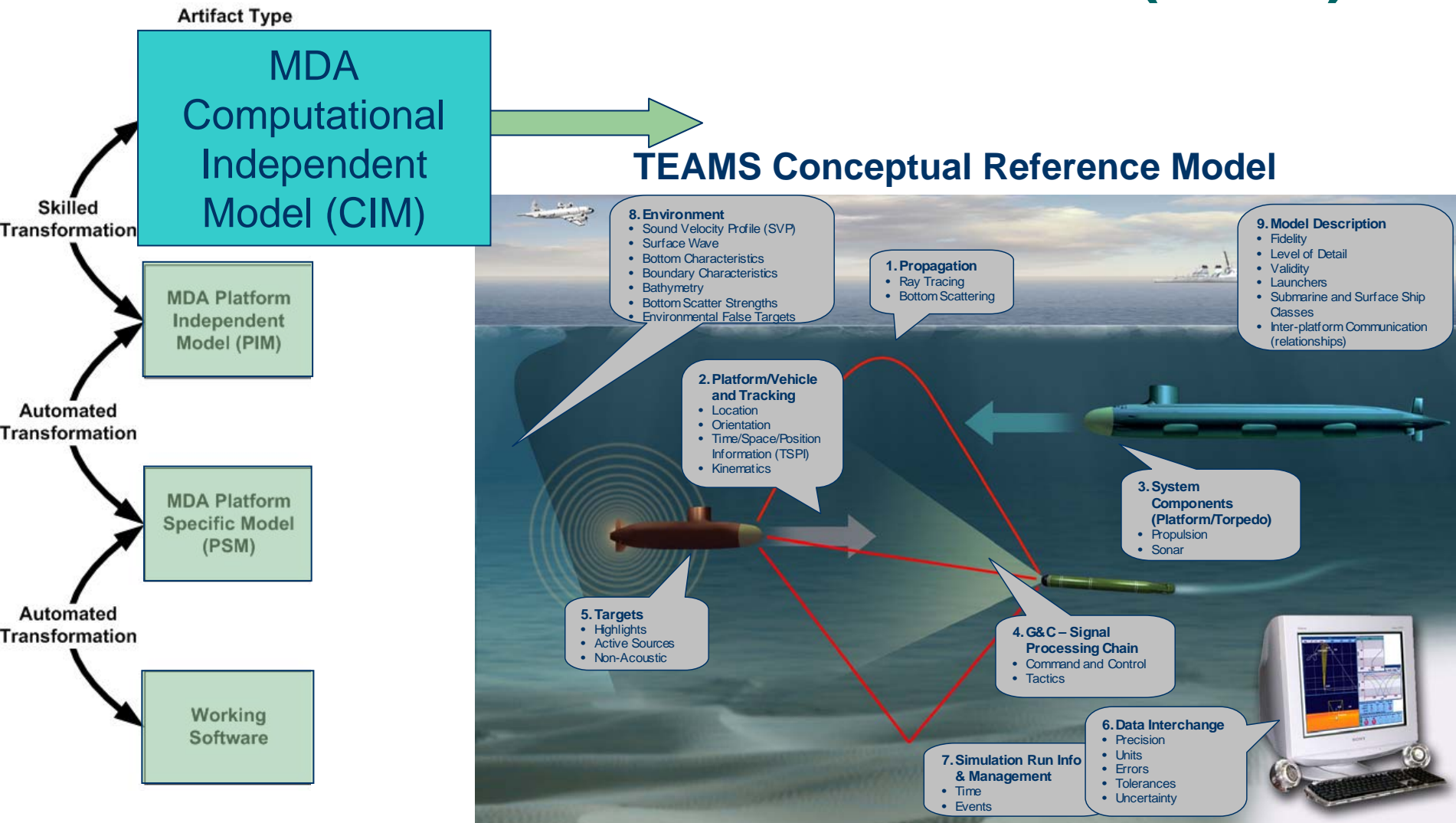


ORBIS Object Examples



Sonar System Toolset (SST)

The Method: Model Driven Architecture (MDA)



TEAMS Conceptual Reference Model

8. Environment

- Sound Velocity Profile (SVP)
- Surface Wave
- Bottom Characteristics
- Boundary Characteristics
- Bathymetry
- Bottom Scatter Strengths
- Environmental False Targets

1. Propagation

- Ray Tracing
- Bottom Scattering

9. Model Description

- Fidelity
- Level of Detail
- Validity
- Launchers
- Submarine and Surface Ship Classes
- Inter-platform Communication (relationships)

2. Platform/Vehicle and Tracking

- Location
- Orientation
- Time/Space/Position Information (TSPI)
- Dynamics

3. System Components (Platform/Torpedo)

- Propulsion
- Sonar

5. Targets

- Highlights
- Active Sources
- Non-Acoustic

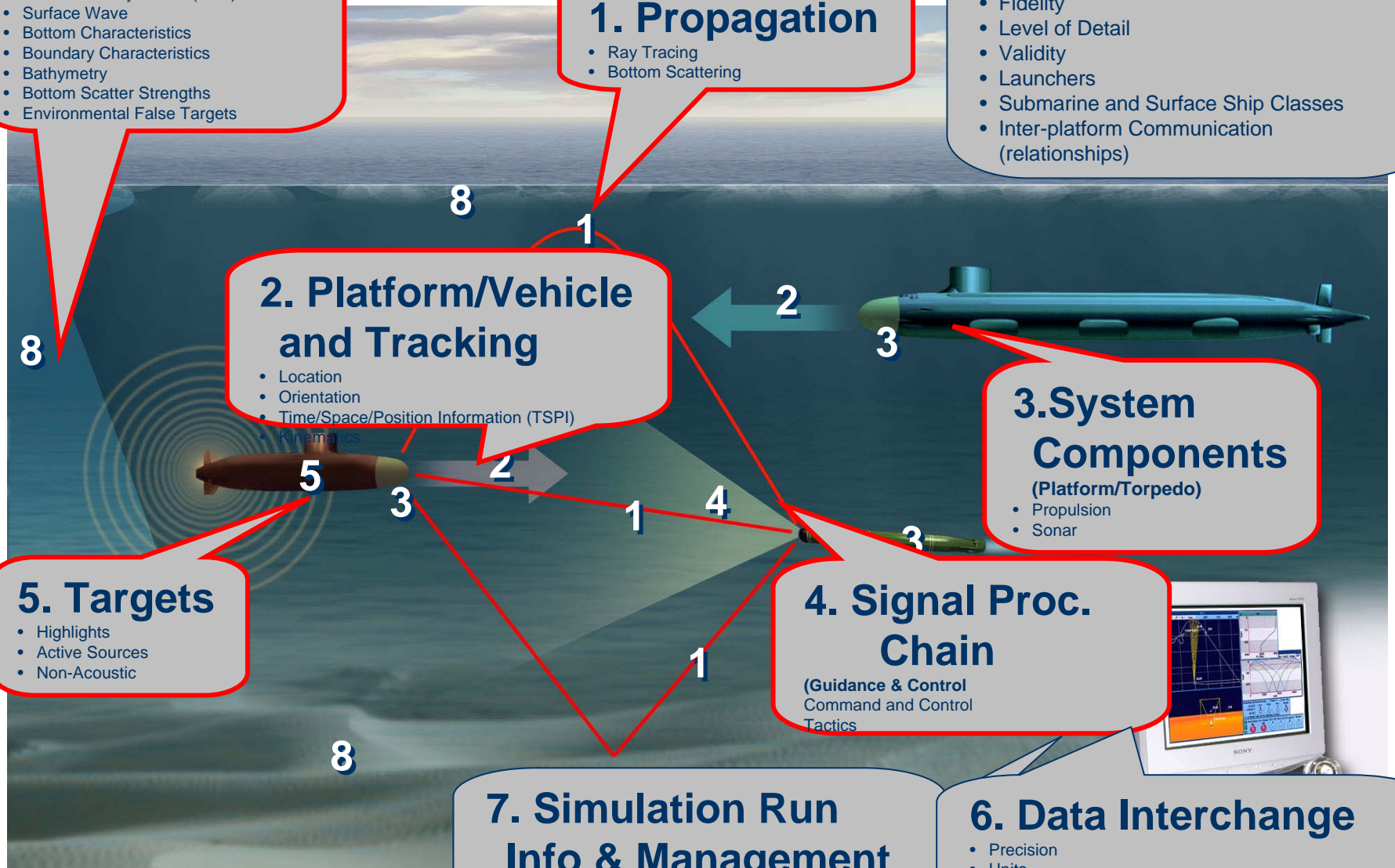
4. Signal Proc. Chain (Guidance & Control Command and Control Tactics)

7. Simulation Run Info & Management

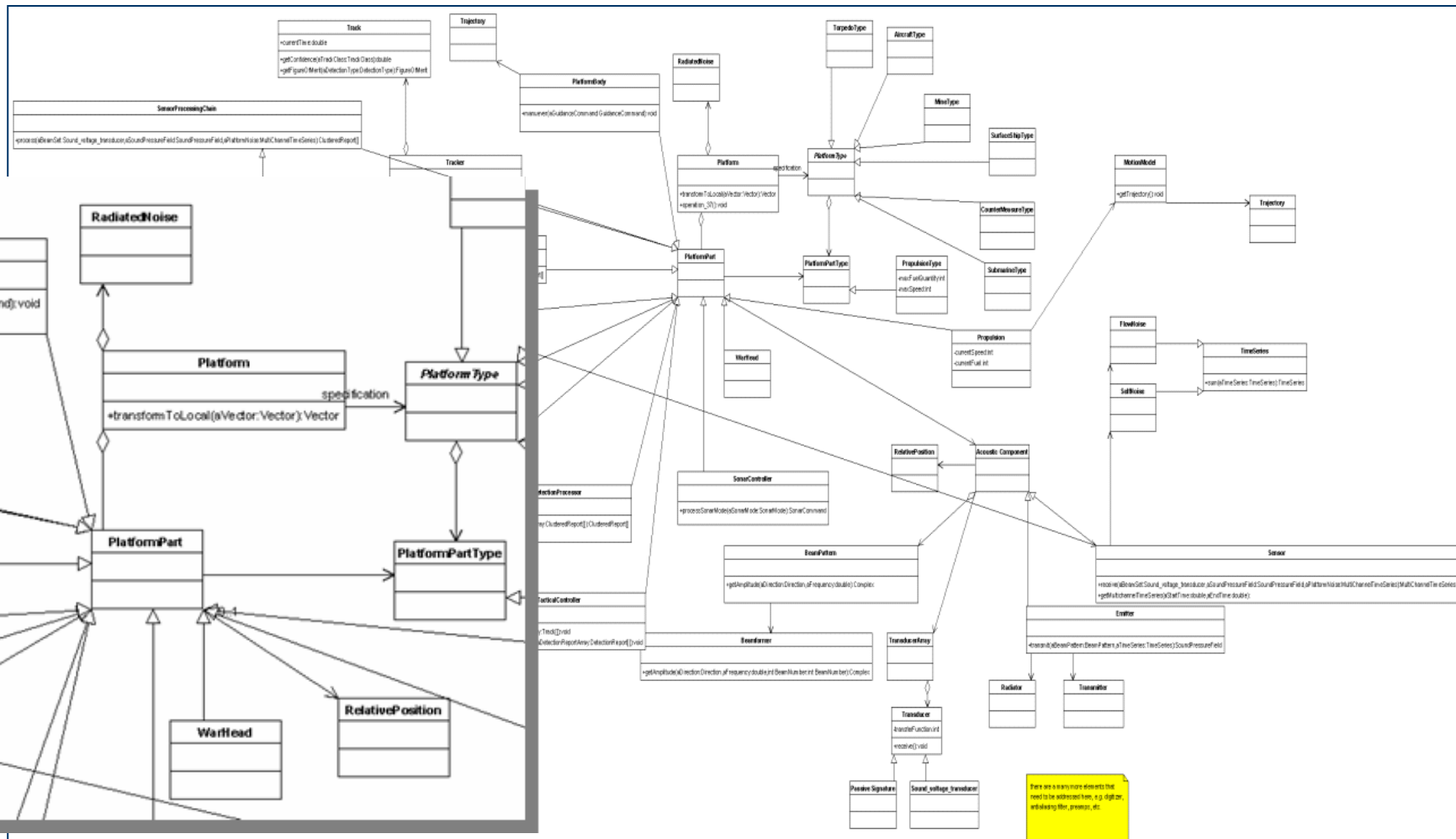
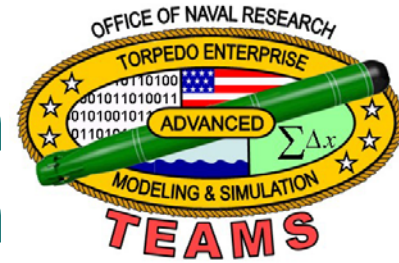
- Time
- Events

6. Data Interchange

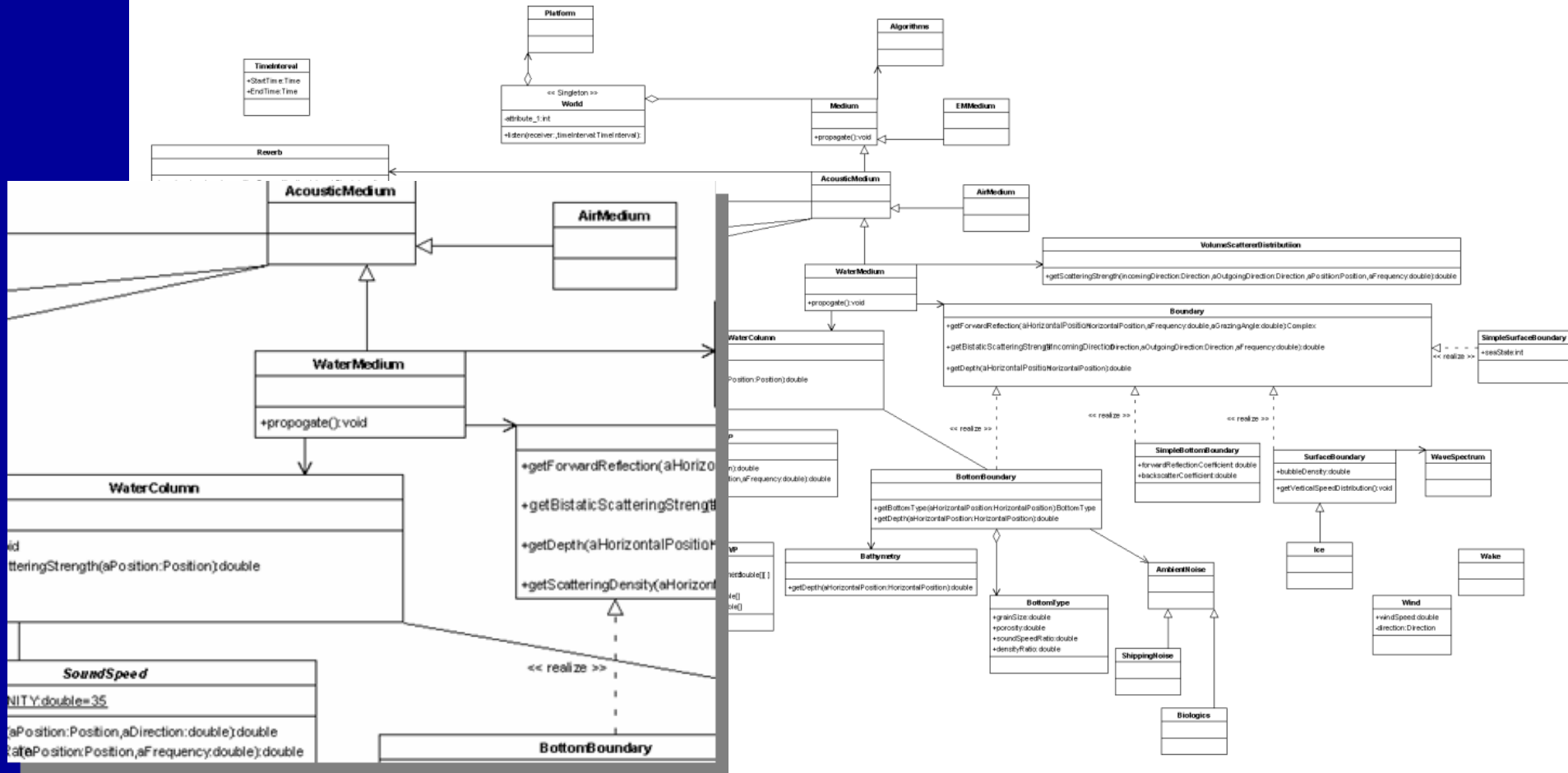
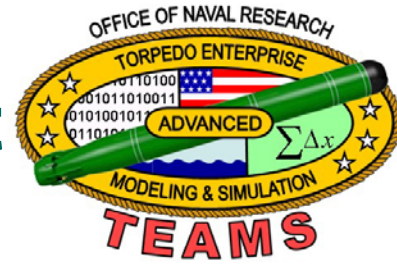
- Precision
- Units
- Errors
- Tolerances
- Uncertainty



Platform Conceptual Level Diagram



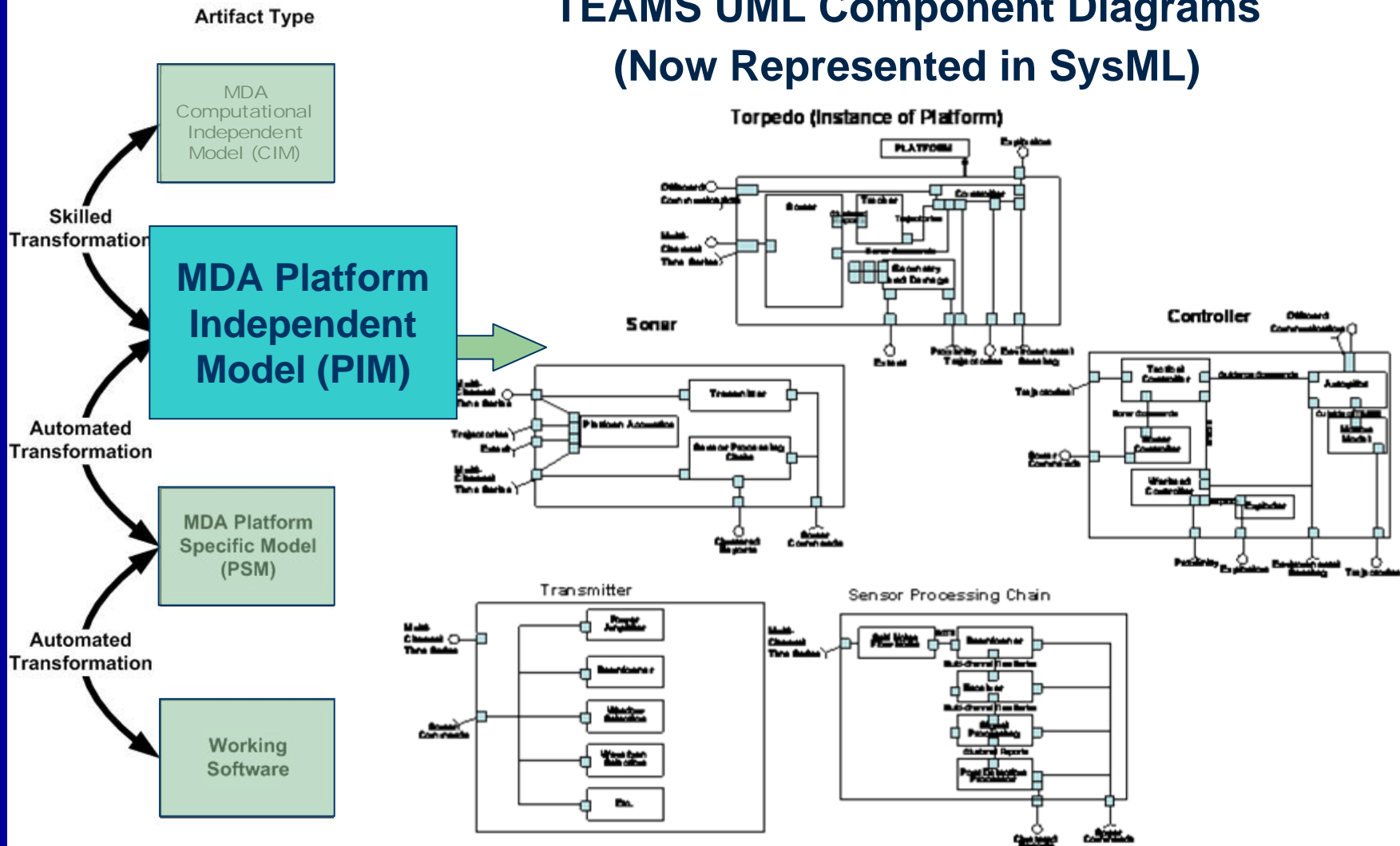
Environment Conceptual Level Diagram



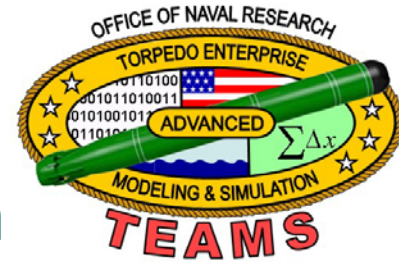
The Method: Model Driven Architecture (MDA)



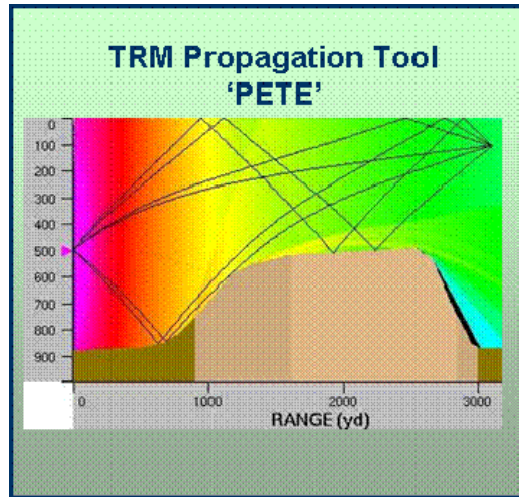
TEAMS UML Component Diagrams (Now Represented in SysML)



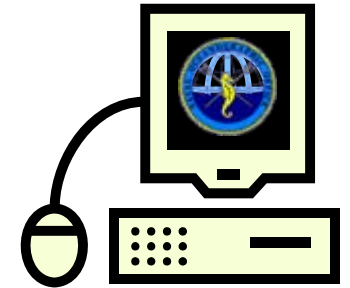
TEAMS PSM: Implementation



Reference Implementations



In-situ Environmental Data via Web Services



NAVOCEANO SIPRNET Web Site

Jackson Bottom Model via CORBA

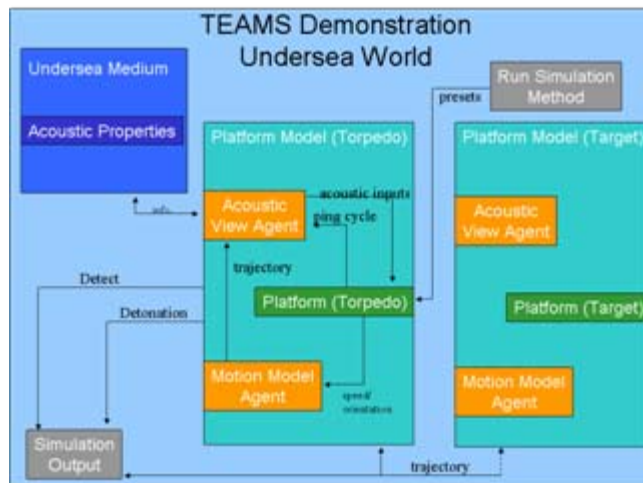


Applied Physics Lab
University of Washington

Closed-Loop SimuLink™ Torpedo, Environment & Target



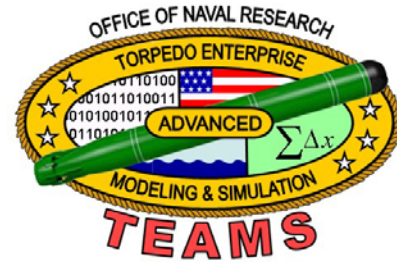
PMS 404
UNDERSEA WEAPONS PROGRAM OFFICE





TEAMS SysML Proof of Concept

- **Port existing UML to SysML**
 - Torpedo system components
 - Simulation environment
- **Extend TEAMS SysML to include:**
 - Requirements traceability
 - Parametrics and constraints
- **Share experiences and lessons learned using SysML for architecture and component modeling**



UML to SysML Approach

- Convert UML Class Diagrams to SysML Block Definition Diagrams (BDDs)
- Convert UML Component Diagrams to SysML Internal Block Diagrams (IBDs)
- Represent Behavior Relationships Between Blocks as Activity Diagrams (*new!*)
- Capture Requirements Traceability (*new!*)
- Capture Parametric Relationships and Constraints (*new!*)

TEAMS Perspective: SysML Pros and Cons



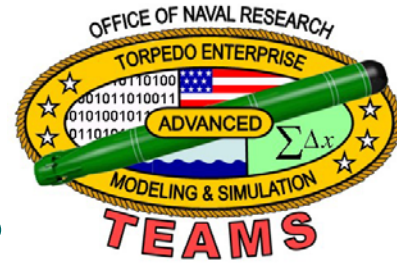
Pros

- **Requirements**
 - Explicitly lay out requirements and consequences
- **Views and Viewpoints**
 - Can separate requirements and model views based on stakeholders concerns
- **Structure**
 - Ability for model structure to verify requirements
 - Can search for requirements that aren't verified
 - Can search for model components that aren't justified
 - Separation of structure from behavior
 - SysML BDDs vs. IBDs and Activities allow for clear separation
 - UML allows this, but easier to implement in SysML
- **Behavior**
 - Dashed line for activity flow is more aesthetically pleasing
 - vs. UML solid line

Cons

- **Allocating CIM to PIM**
 - Difficulty with abstract activities
 - Exit path dependent on logic within an activity is not accessible and can't be modeled
 - Not represented well in either UML or SysML – tactical controller example
- **Implementing PIM**
 - Not “direct” for some SysML features
 - Flow ports, continuous activities, parametric constraints involve more components than just themselves
 - Flows in “real systems” easier to represent
 - Flows in software modeling are open to interpretation
 - Requires additional documentation of model to bridge between SysML feature and executable code

TEAMS Perspective: SysML Pros



Pros

- **Requirements**
 - Explicitly lay out requirements and consequences
- **Views and Viewpoints**
 - Can separate requirements and model views based on stakeholders concerns
- **Structure**
 - Ability for model structure to verify requirements
 - Can search for requirements that aren't verified
 - Can search for model components that aren't justified
 - Separation of structure from behavior
 - SysML BDDs vs. IBDs and Activities allow for clear separation
 - UML allows this, but easier to implement in SysML
- **Behavior**
 - Dashed line for activity flow is more aesthetically pleasing
 - vs. UML solid line



Sponsor Requirements

req SponsorRequirements [SponsorReqts]

Reduced Duplicate Efforts

notes

Different contractors should not have to research the same technology or enabling model in order to accomplish their specific goals. Instead, similar efforts should be merged together and the result shared.

Less Component Integration Time

notes

Component developers should be able to spend their time and resources on developing, and be able to verify new ideas with simulation quickly.

Model Realizable Systems

notes

Component developments need to be convertible into a real system to be useful.

Contractor Interoperability

notes

If two different contractors write two different components, they should be able to communicate with each other.

Reuse Legacy and New Components

notes

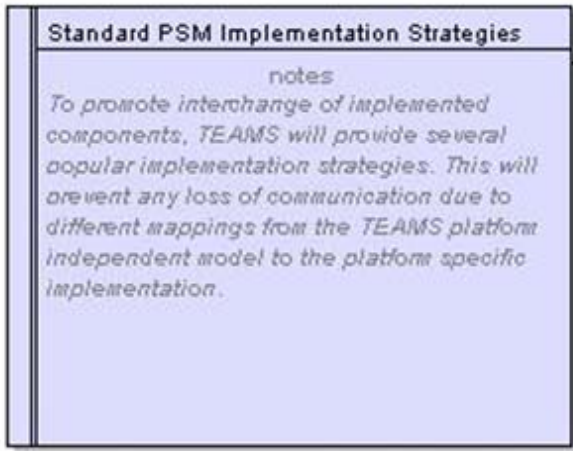
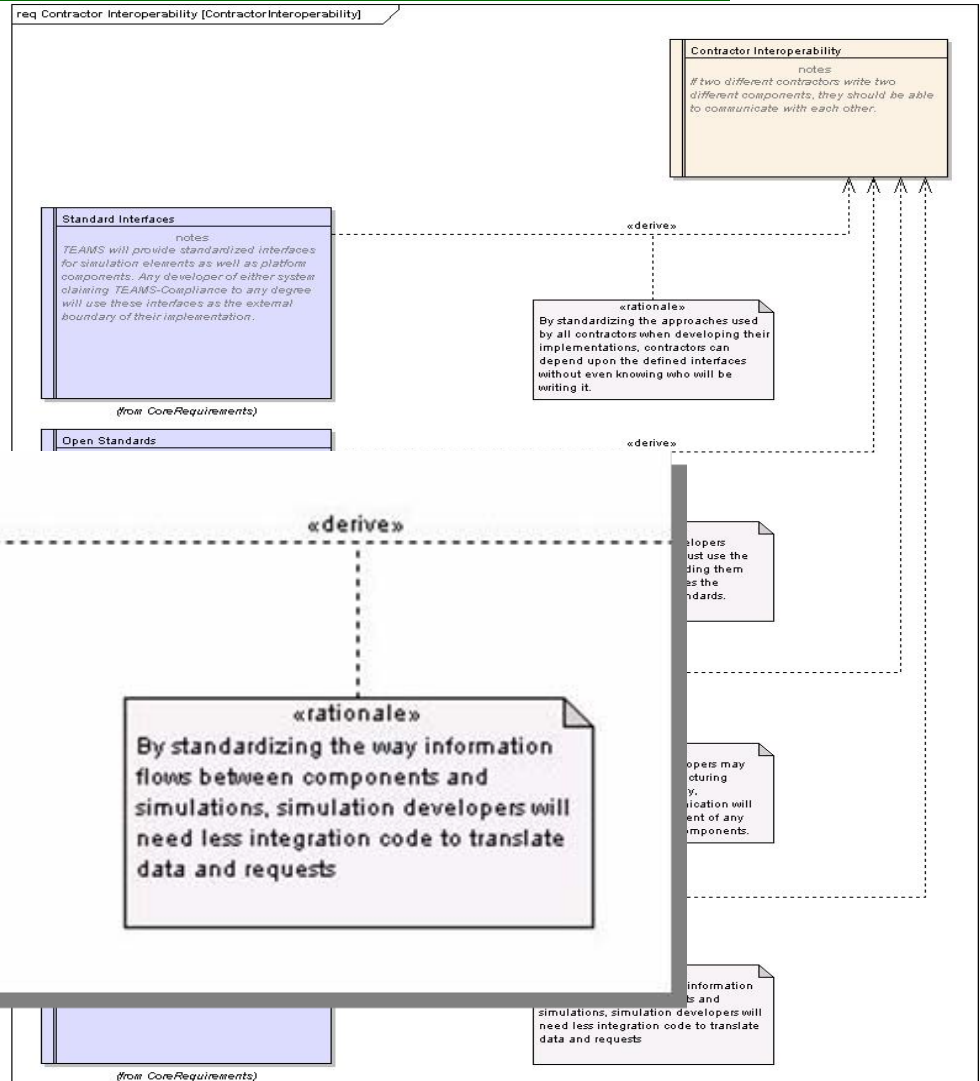
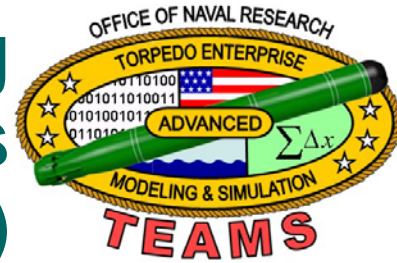
Some mechanism should enable older systems to be pulled into simulations with new interfaces, and newly developed components should have some easily reusable interface to reduce this problem in the future.

Room for Future Growth

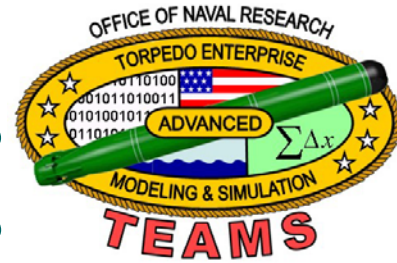
notes

Adaptivity to future changes is important in any large initial investment, including standardization of components. There is a risk of the standards being out of date before they have enough time to be useful.

Rationale for Deriving TEAMS Core Values from Sponsor Requirement(s)



Requirements Traceability: TEAMS Core Values



Standard Interfaces

notes

TEAMS will provide standardized interfaces for simulation elements as well as platform components. Any developer of either system claiming TEAMS-Compliance to any degree will use these interfaces as the external boundary of their implementation.

Model Realizable Systems

notes

The interfaces that appear in the TEAMS model will reflect actual systems in the real world. This includes designed systems as well as physical constraints placed by the environment.

req CoreRequirements [CoreRequirements]

Standard Interfaces

notes

TEAMS will provide standardized interfaces for simulation elements as well as platform components. Any developer of either system claiming TEAMS-Compliance to any degree will use these interfaces as the external boundary of their implementation.

Platform Independence

notes

The interfaces that TEAMS provides will not preclude any particular implementation platform in their design. This includes but is not limited to considerations such as transport mechanism, operating system, and programming language.

Open Standards

notes

To promote the usefulness of standardized interfaces, TEAMS must make those interfaces public and available to any interested party.

Model Realizable Systems

notes

The interfaces that appear in the TEAMS model will reflect actual systems in the real world. This includes designed systems as well as physical constraints placed by the environment.

Extensible Interfaces

notes

The TEAMS interfaces will not be binding contracts of behavior, but rather a basis of communication between components. These interfaces will be extendable to include new ways of communication, and new behaviors of established communications.

Evolving Standards

notes

TEAMS will update their model periodically whenever such changes are required to preserve an up-to-date reflection of actual systems.

Loosely Coupled Interfaces

notes

TEAMS will design the interfaces such that they do not depend on the internal structure of any other interface and, where possible, do not depend on the existence of another interface at all.

Tiers of Interfaces

notes

The interface model will layer its interfaces such that higher levels completely compose lower levels, and no interface or behavior ever depends on the lower structure of an interface in its own tier. Communication between components will only exist within the same interface tier, or when communicating with the component's parent or child component.

Support Different Levels of Detail

notes

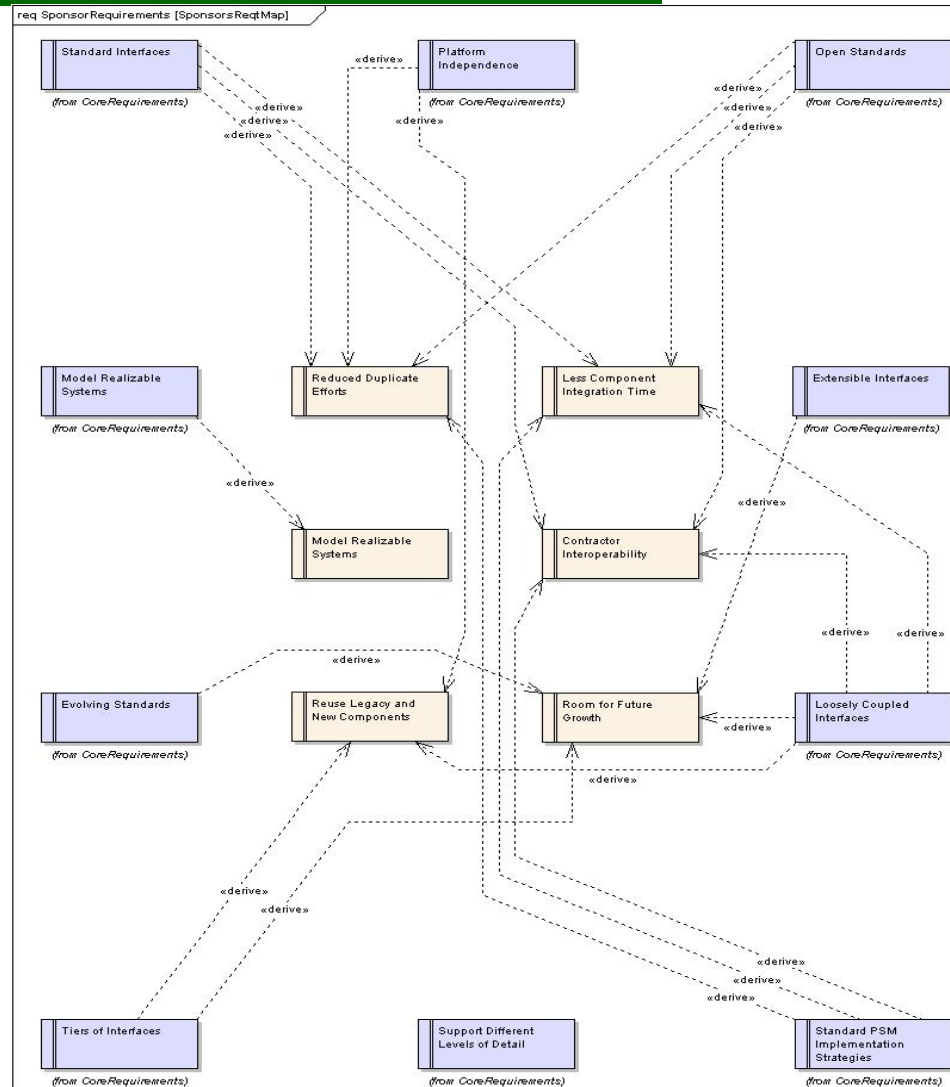
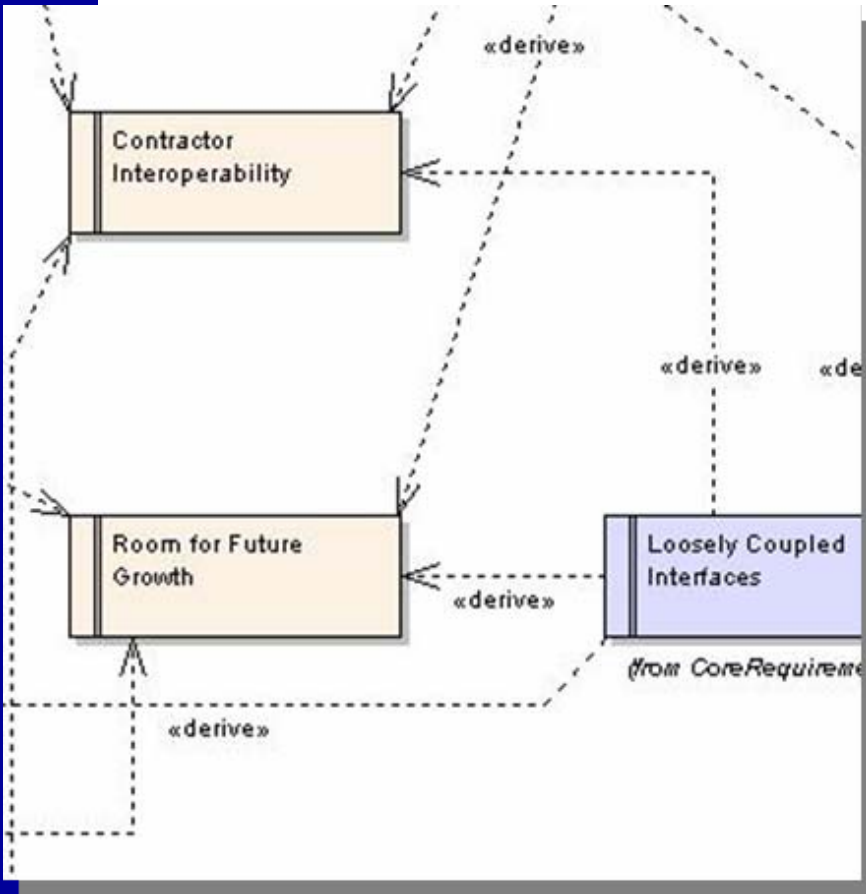
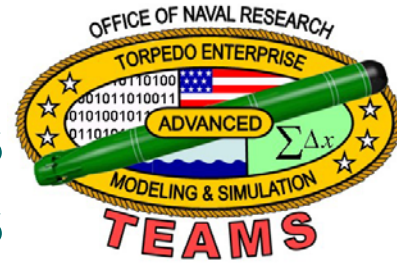
Where possible, TEAMS will use value types abstractly to avoid specifying the level of detail present.

Standard PSM Implementation Strategies

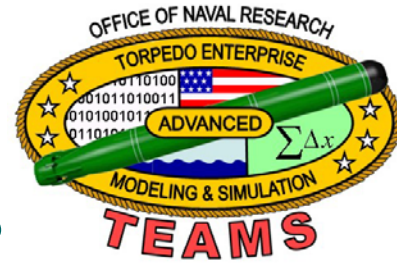
notes

To promote interchange of implemented components, TEAMS will provide several popular implementation strategies. This will prevent any loss of communication due to different mappings from the TEAMS platform independent model to the platform specific implementation.

Sponsor Requirements Mapped to TEAMS Core Values



TEAMS Perspective: SysML Pros

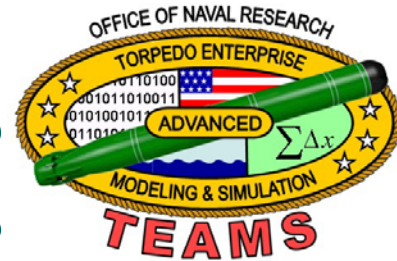


Pros

- Requirements
 - Explicitly lay out requirements and consequences
- Views and Viewpoints
 - Can separate requirements and model views based on stakeholders concerns
- Structure
 - Ability for model structure to verify requirements
 - Can search for requirements that aren't verified
 - Can search for model components that aren't justified
 - Separation of structure from behavior
 - SysML BDDs vs. IBDs and Activities allow for clear separation
 - UML allows this, but easier to implement in SysML
- Behavior
 - Dashed line for activity flow is more aesthetically pleasing
 - vs. UML solid line

TEAMS

Stakeholder Requirements



req Requirements [Requirements]

«view»
SponsorRequirements

- + Contractor Interoperability
- + Less Component Integration Time
- + Model Realizable Systems
- + Reduced Duplicate Efforts
- + Reuse Legacy and New Components
- + Room for Future Growth

«view»
SimulationDeveloperRequirements

- + Continue using Legacy Systems
- + Design Flexibility
- + Easier Maintenance and Upgrades
- + Less Component Integration Time
- + Simulate Real Situations

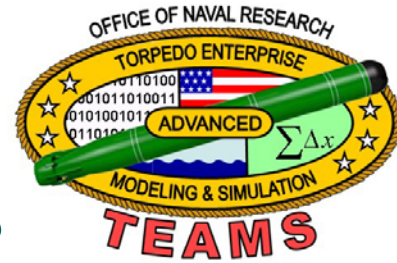
«view»
ComponentDeveloperRequirements

- + Design Flexibility
- + Develop New Approaches
- + Intellectual Property Rights
- + *Less Component Integration Time*
- + Model Real Components
- + Scalable Component Design
- + Simulation Interoperability

«view»
FleetRequirements

- + Better Systems
- + Commonality
- + Highly Detailed Simulations
- + Shorter Acquisition Period

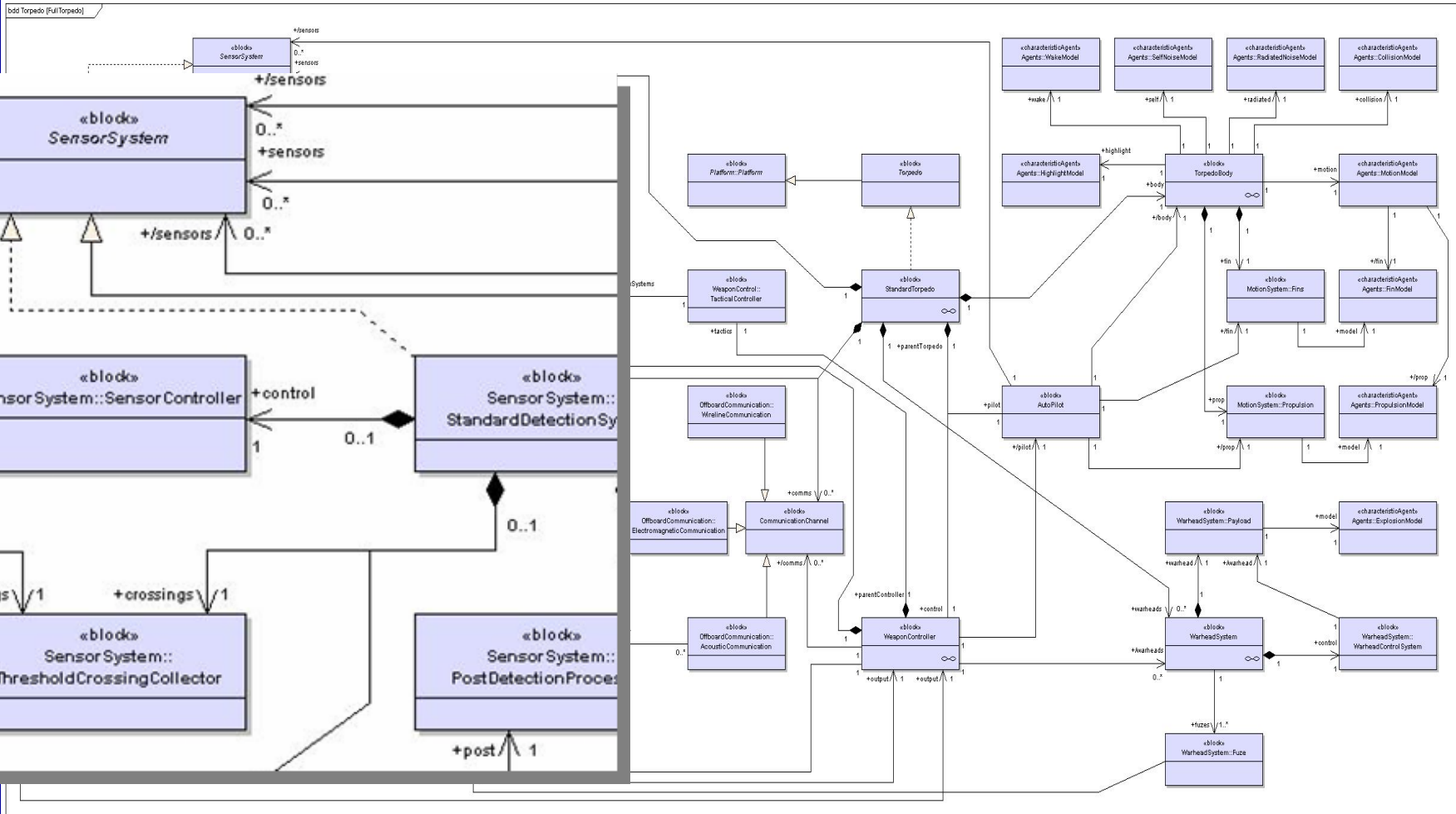
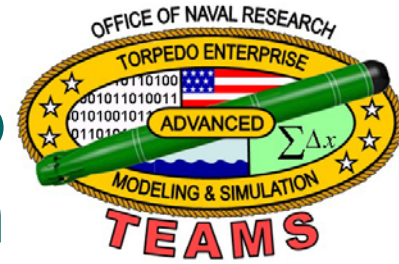
TEAMS Perspective: SysML Pros



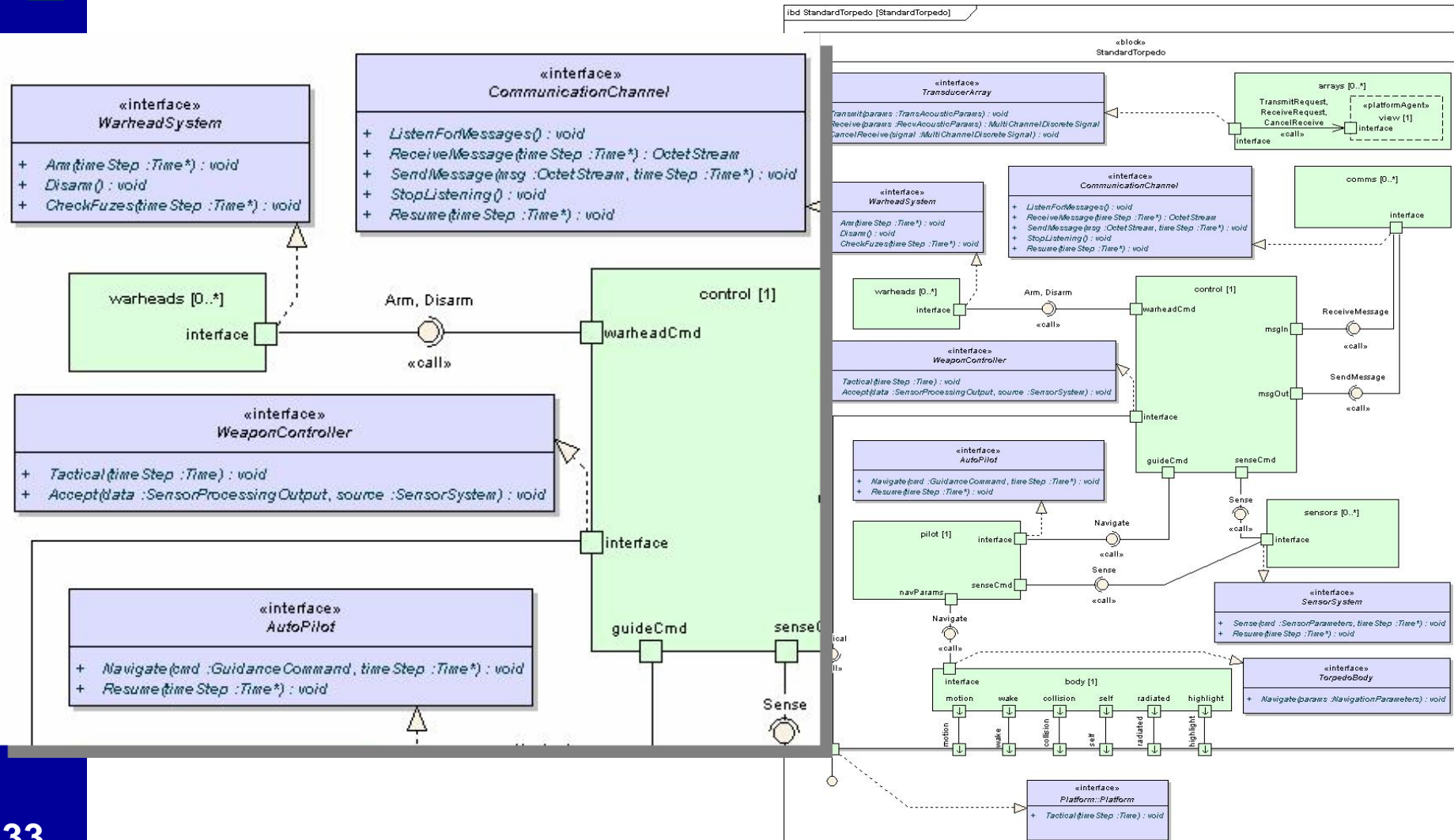
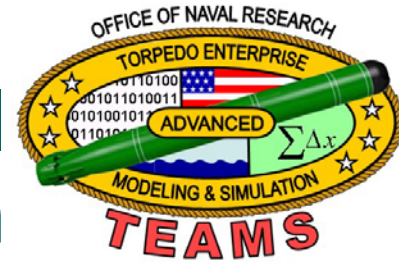
Pros

- **Requirements**
 - Explicitly lay out requirements and consequences
- **Views and Viewpoints**
 - Can separate requirements and model views based on stakeholders concerns
- **Structure**
 - Ability for model structure to verify requirements
 - Can search for requirements that aren't verified
 - Can search for model components that aren't justified
 - Separation of structure from behavior
 - SysML BDDs vs. IBDs and Activities allow for clear separation
 - UML allows this, but easier to implement in SysML
- **Behavior**
 - Dashed line for activity flow is more aesthetically pleasing
 - vs. UML solid line

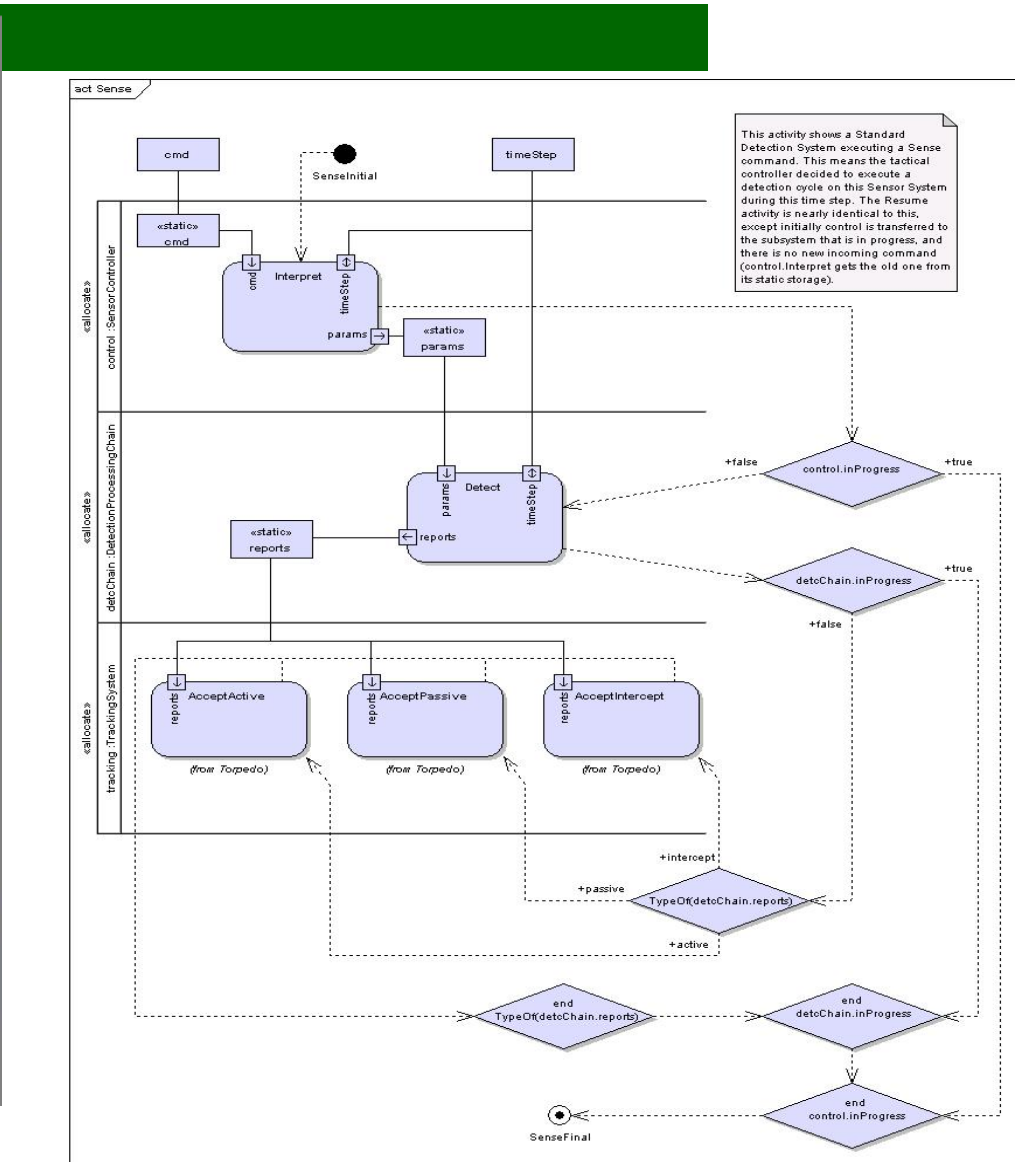
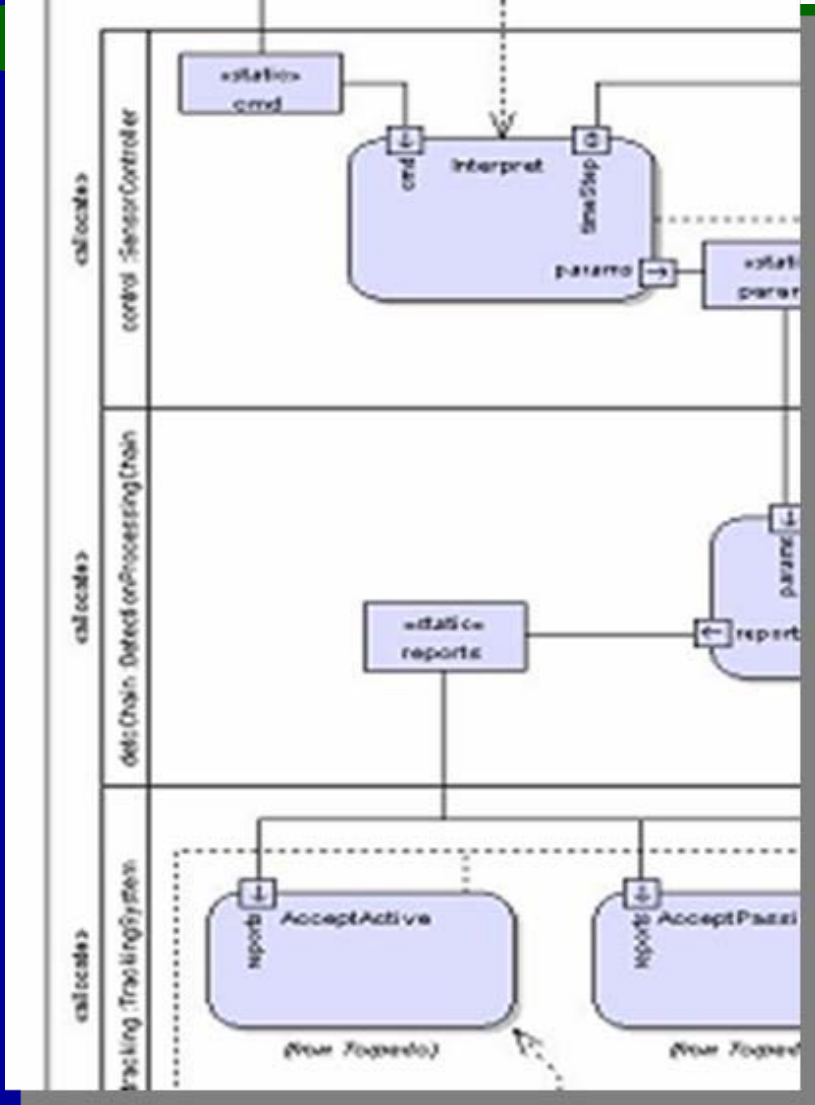
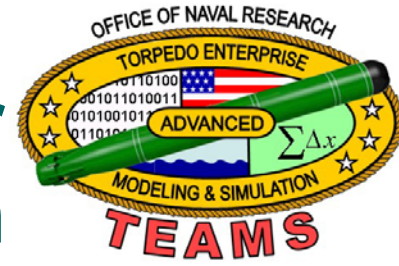
Torpedo Block Definition Diagram



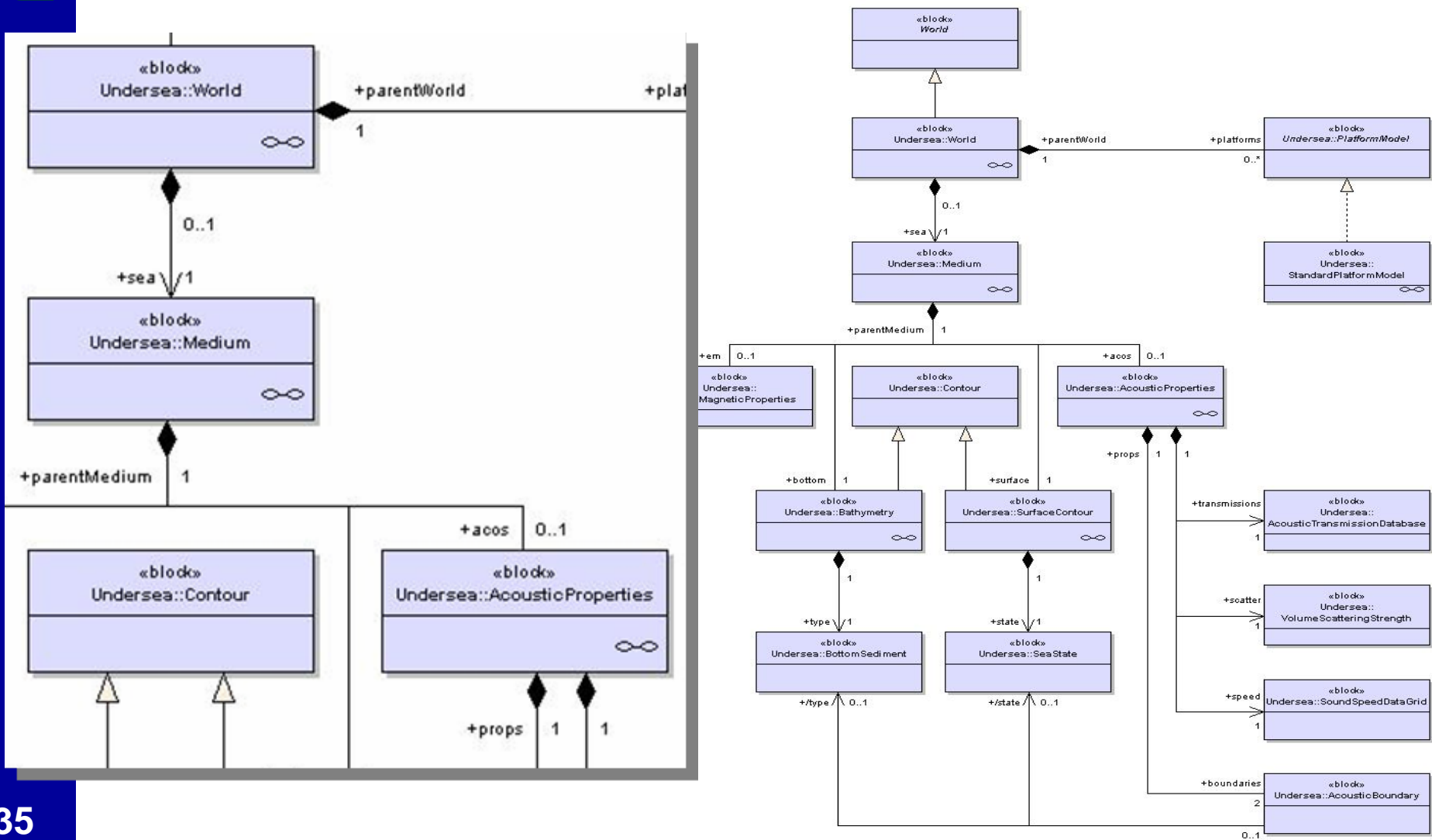
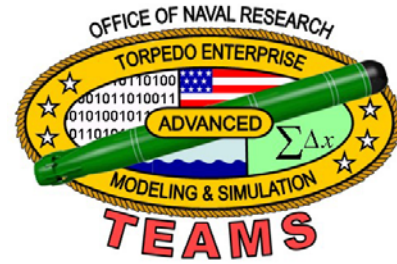
Torpedo Internal Block Definition Diagram



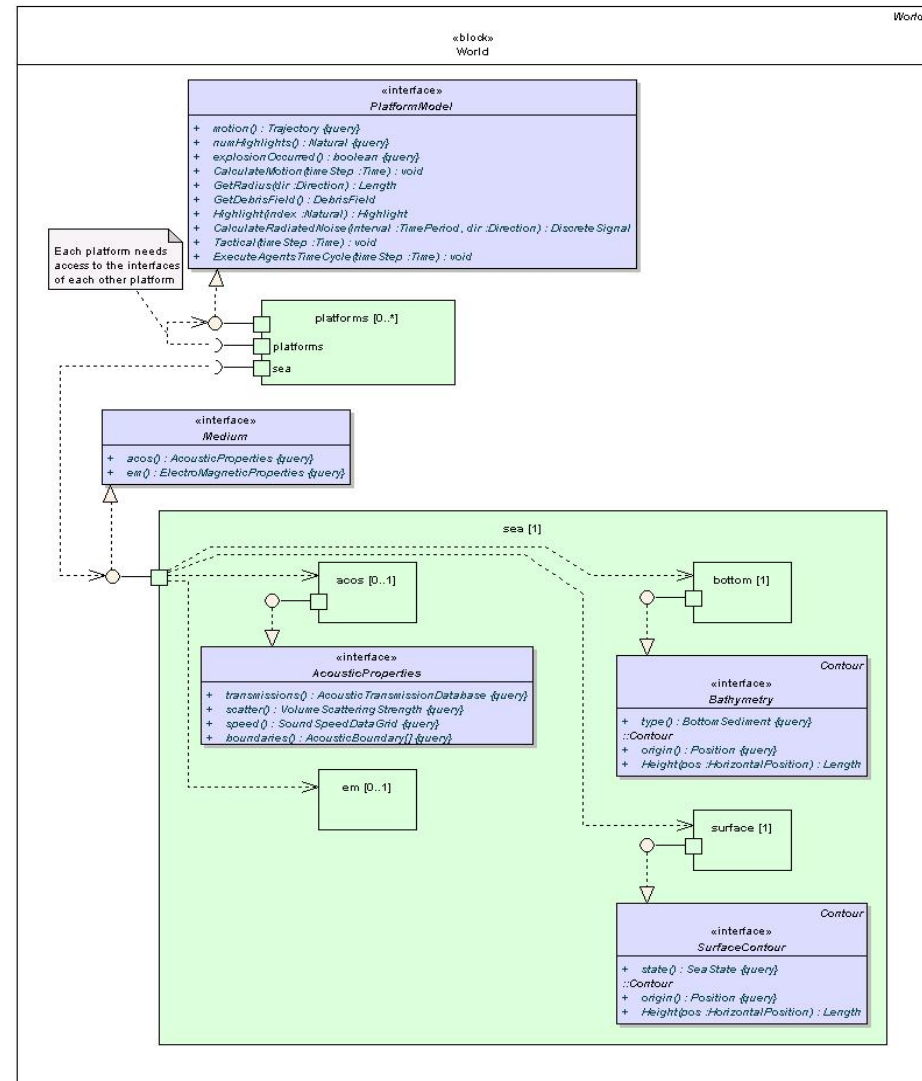
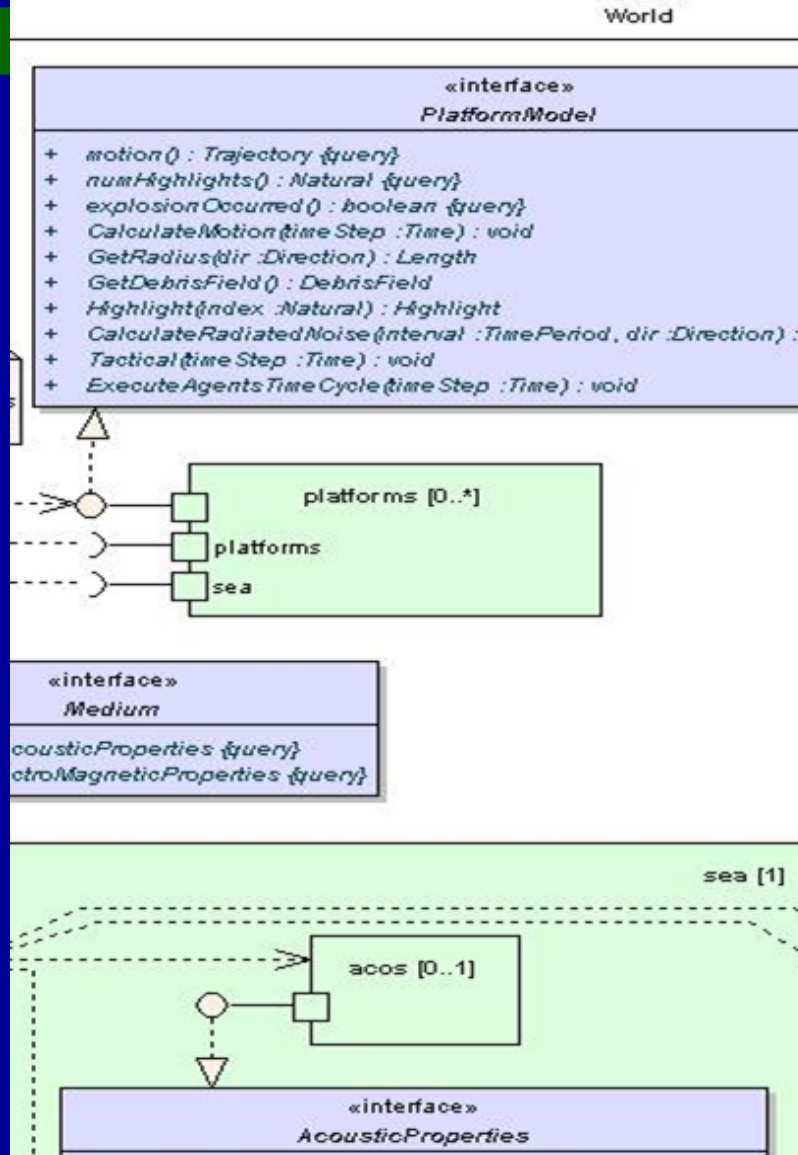
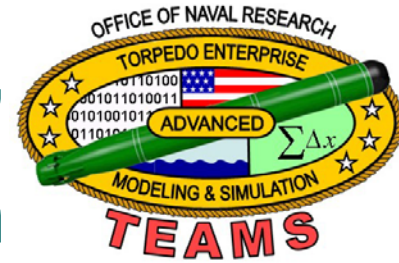
Torpedo Sensor Activity Diagram



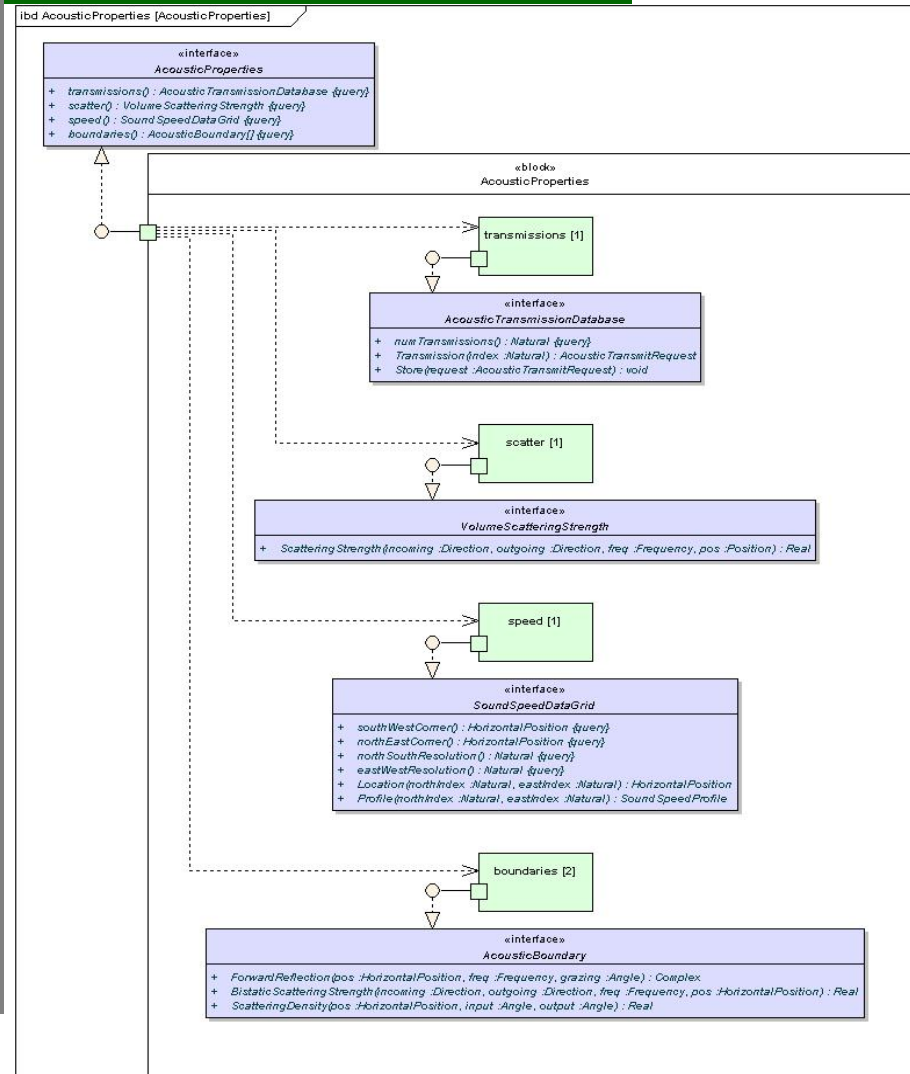
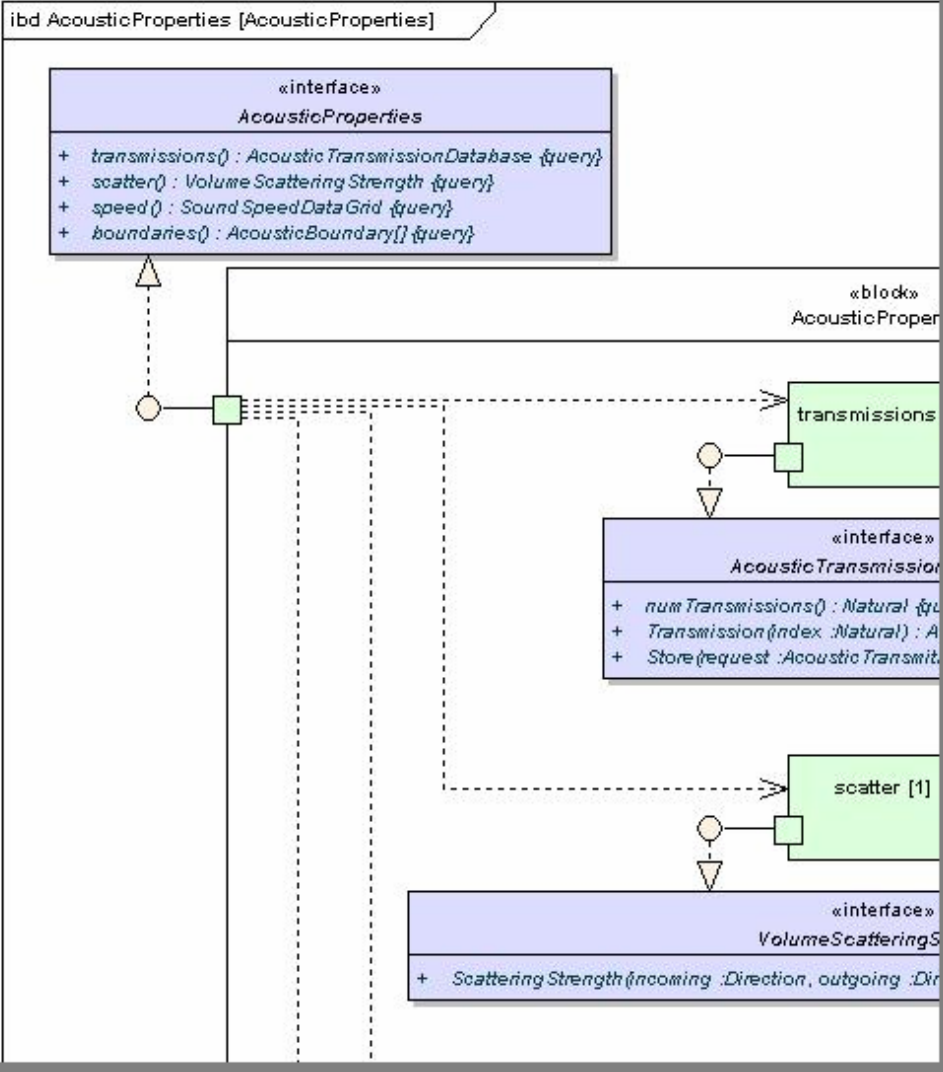
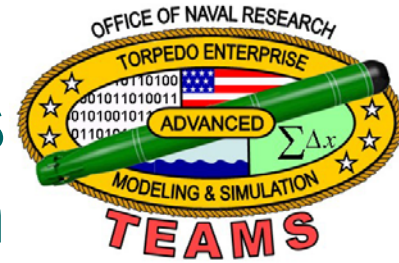
Undersea World Block Definition Diagram



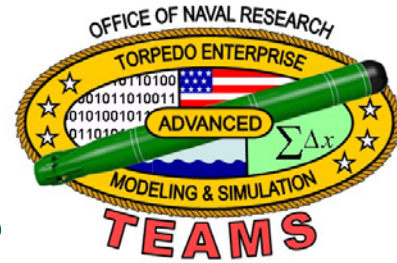
Simulation "World" Internal Block Definition Diagram



Acoustic Properties Internal Block Definition Diagram



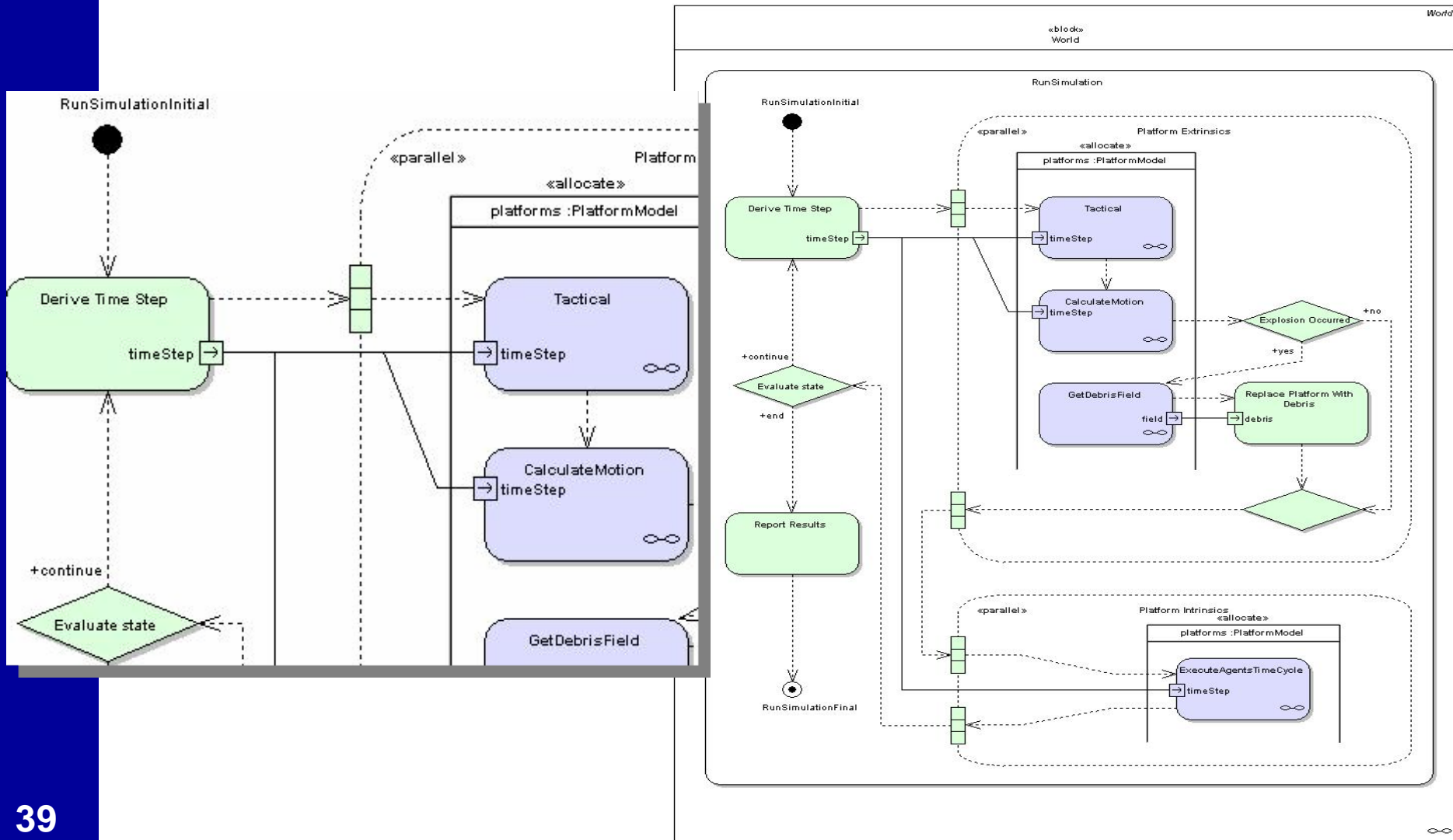
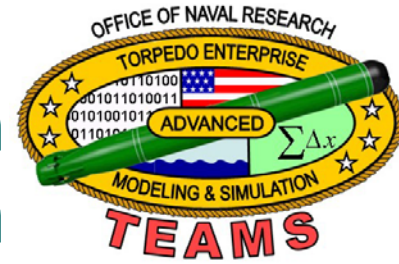
TEAMS Perspective: SysML Pros



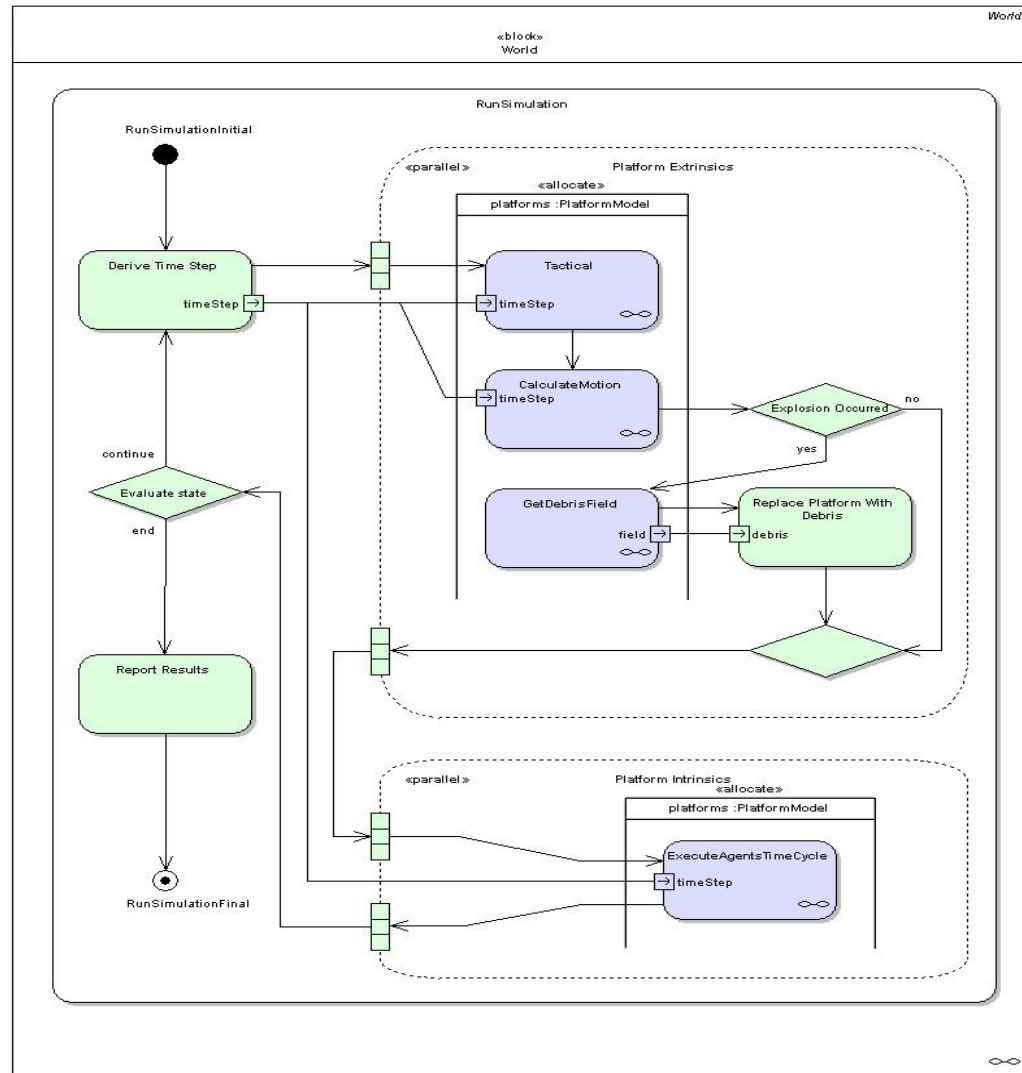
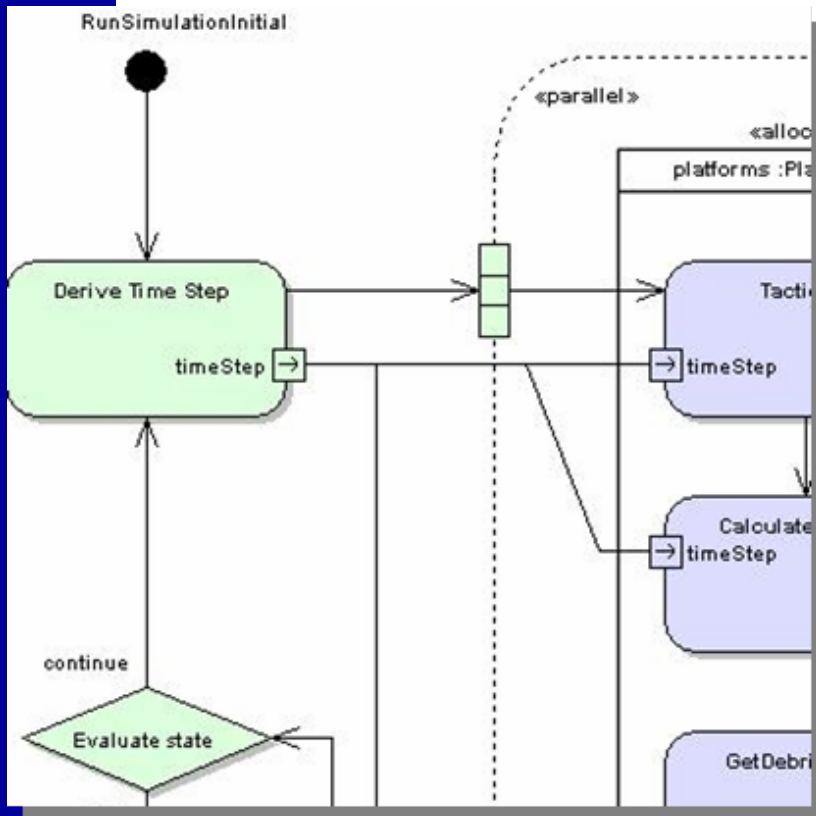
Pros

- **Requirements**
 - Explicitly lay out requirements and consequences
- **Views and Viewpoints**
 - Can separate requirements and model views based on stakeholders concerns
- **Structure**
 - Ability for model structure to verify requirements
 - Can search for requirements that aren't verified
 - Can search for model components that aren't justified
 - Separation of structure from behavior
 - SysML BDDs vs. IBDs and Activities allow for clear separation
 - UML allows this, but easier to implement in SysML
- **Behavior**
 - Dashed line for activity flow is more aesthetically pleasing
 - vs. UML solid line

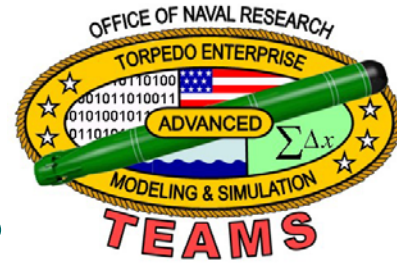
Simulation “World” Activity Diagram



Solid Line Representation



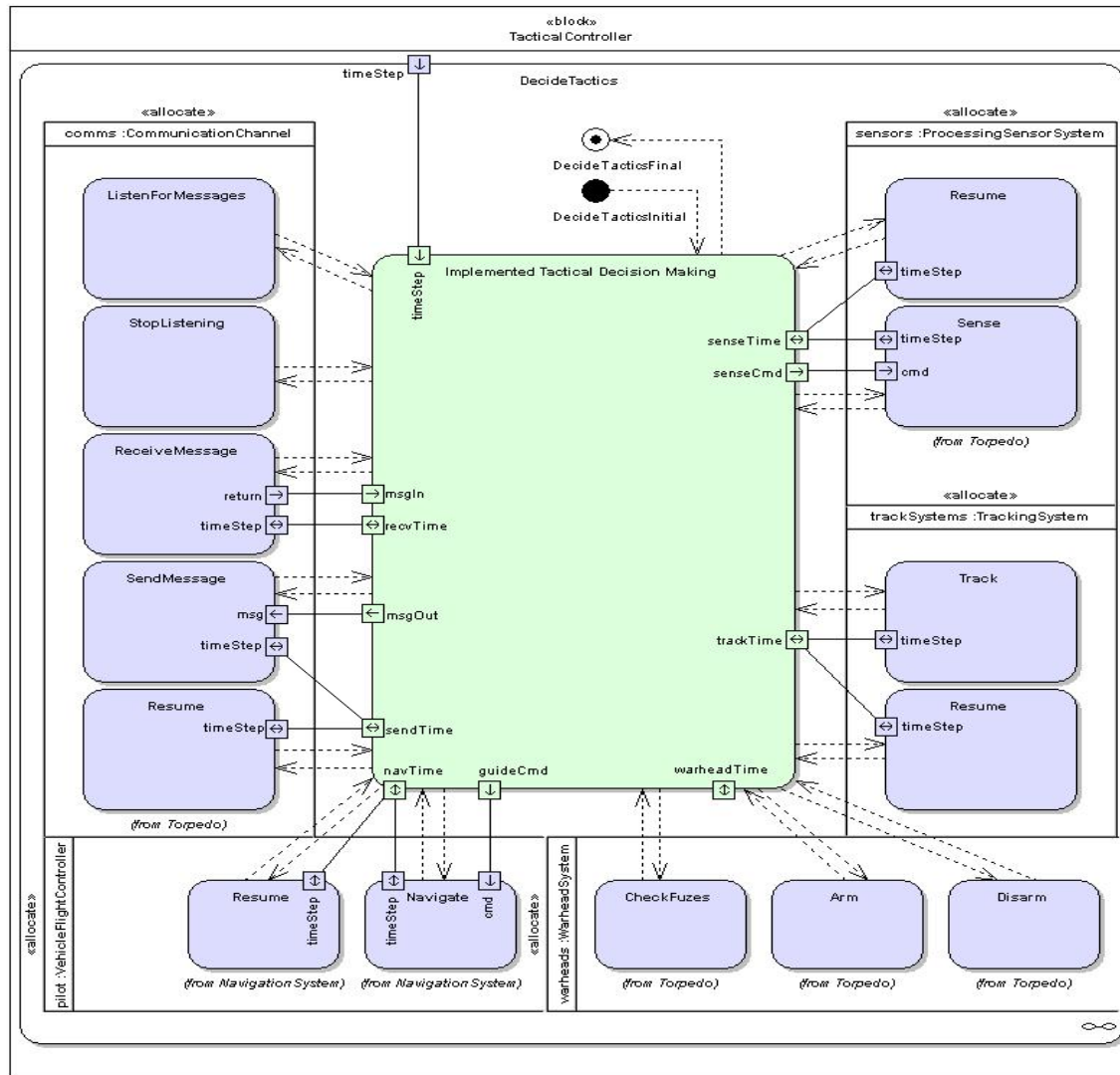
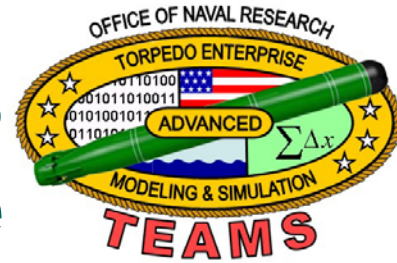
TEAMS Perspective: SysML Cons



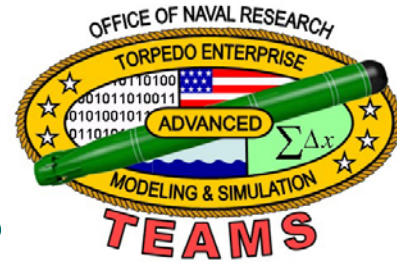
Cons

- **Allocating CIM to PIM**
 - Difficulty with abstract activities
 - Exit path dependent on logic within an activity is not accessible and can't be modeled
 - Not represented well in either UML or SysML – tactical controller example
- **Implementing PIM**
 - Not “direct” for some SysML features
 - Flow ports, continuous activities, parametric constraints involve more components than just themselves
 - Flows in “real systems” easier to represent
 - Flows in software modeling are open to interpretation
 - Requires additional documentation of model to bridge between SysML feature and executable code

TEAMS Tactical Controller Example



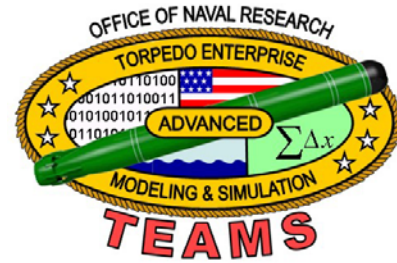
TEAMS Perspective: SysML Cons



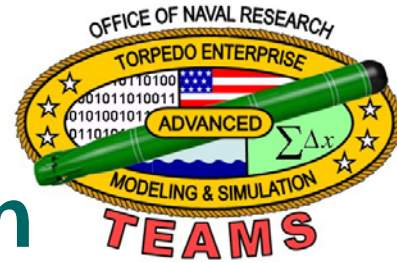
Cons

- **Allocating CIM to PIM**
 - **Difficulty with abstract activities**
 - Exit path dependent on logic within an activity is not accessible and can't be modeled
 - Not represented well in either UML or SysML – tactical controller example
- **Implementing PIM**
 - **Not “direct” for some SysML features**
 - Flow ports, continuous activities, parametric constraints involve more components than just themselves
 - Flows in “real systems” easier to represent than simulations
 - Flows in software modeling are open to interpretation
 - **Requires additional documentation of model to bridge between SysML feature and executable code**

Lessons Learned and Value Added



- Requirements traceability is vital to the success of several TEAMS projects
 - ONR TEAMS standard framework and interfaces
 - OSD-ATL feasibility study
 - TOGAF/MDA Synergy Project
- SysML was designed with “real” systems in mind
 - where UML is software oriented
- Perceived concreteness – simulated vs. actual system
 - not just one way to design interfaces, need recommendations for implementation
- Still need some UML features not present in SysML
 - <<Instantiate>> or <<create>> for dynamic allocation
- Still need guidance on how to best implement parametrics and constraints for modeling and simulation



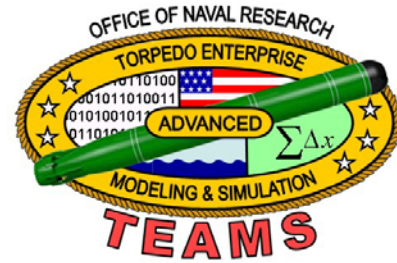
OMG SE DSIG Recommendation

“Clarify the distinction between the domain model and the simulation design model.”

*Reference SE DSIG minutes from OMG San Diego Meeting on March 27, 2007

Integrating SysML Models with Simulation Models

- Goal
 - Integrate system design models with simulation and analysis models
 - Use SysML models to specify an executable architecture
 - Use simulation and analysis models to analyze performance
- How can they work together ?
 - Plug the SysML executable architecture model into a simulation infrastructure to establish a dynamic interface
 - Use the executable architecture model to control the sequence of activities (e.g. detect target, launch weapon)
 - Use the simulation model to compute the parameter values (e.g. missile range to target vs. time)
- What is needed?
 - Approach to use SysML architectural model to specify simulation requirements (use of parametrics?)
 - Harmonization between SysML and simulation standards (i.e. HLA) ?



Future Direction

Working to Establish an Activity for SysML / Simulation Integration Approach

- Formulation/establishment during INCOSE MBSE Workshop in Albuquerque on January 24-25
- Liaison to the INCOSE Model Base Systems Engineering (MBSE) Initiative
- Keep abreast of industry related activities
- Help to foster interaction in this area across industry, government and academia to help move towards the INCOSE MBSE Vision.
- Explore this integration through SISO.



Acknowledgements

- **LtCol Telford / Dwayne Hardy and OSD-ATL; supported and funded this effort for FY07**
- **David Drumheller and ONR; supports and funds TEAMS**
- **Sanford Friedenthal of Lockheed Martin; contributed his expertise and willingness to educate the TEAMS consortium on the nuances of SysML**
- **Members of The Open Group, Object Management Group, and TEAMS Consortium; contributed to the success of SysML Project**
- **Sparx Systems; provided complimentary licenses for Enterprise Architect 6.5 for this SysML effort**



References

- Armstrong, C., Cerenzia, J., Harrington, E., Rivett, P., Waskeiwicz, F., , TOGAF/MDA/IC Synergy Project: Integration Proof-of-Concept Results, Proceedings of the Global Information Summit 2007.

Available:

<http://www.omg.org/docs/omg/07-05-01.pdf>

- Cerenzia, J. L., Scrudder, R.; Goddard, R. P., Haley, T. B., Lounsbury, D. M., Practical Experiences in Creating Components from Legacy Simulations, Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (IITSEC), 2005.

Available:

<http://ntsa.metapress.com/app/home/contribution.asp?referrer=parent&backto=issue,111,153;journal,2,7;linkingpublicationresults,1:113340,1>