

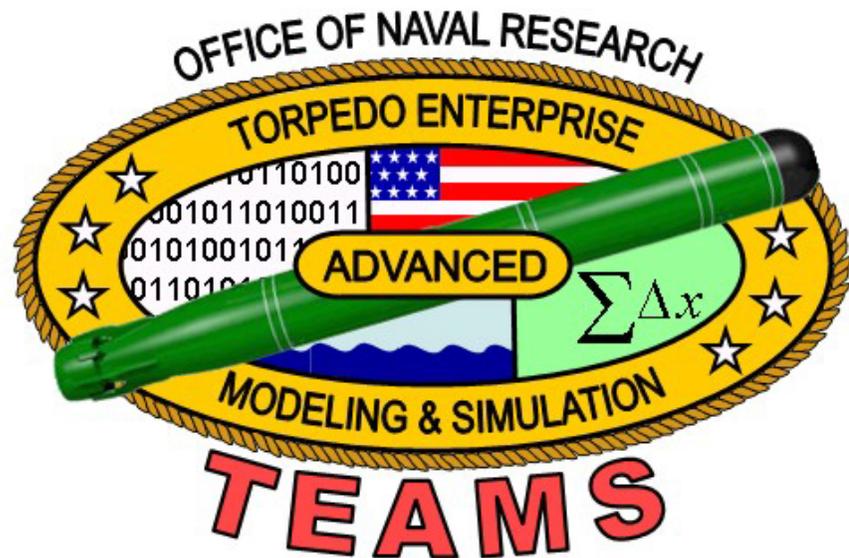
TEAMS and SysML: Proof of Concept Status

Presented to:
Modeling and Simulation Committee of the
National Defense Industrial Association
Systems Engineering Division

Presented by:
Thomas Haley
Naval Undersea Warfare Center

David Diederich
Applied Research Laboratory
Penn State University

17 April 2007



Note: Slides have been updated to incorporate comments from the original presentation -tbh

Outline



- **SysML Case Study Motivation**
- **TEAMS Background**
- **TEAMS 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



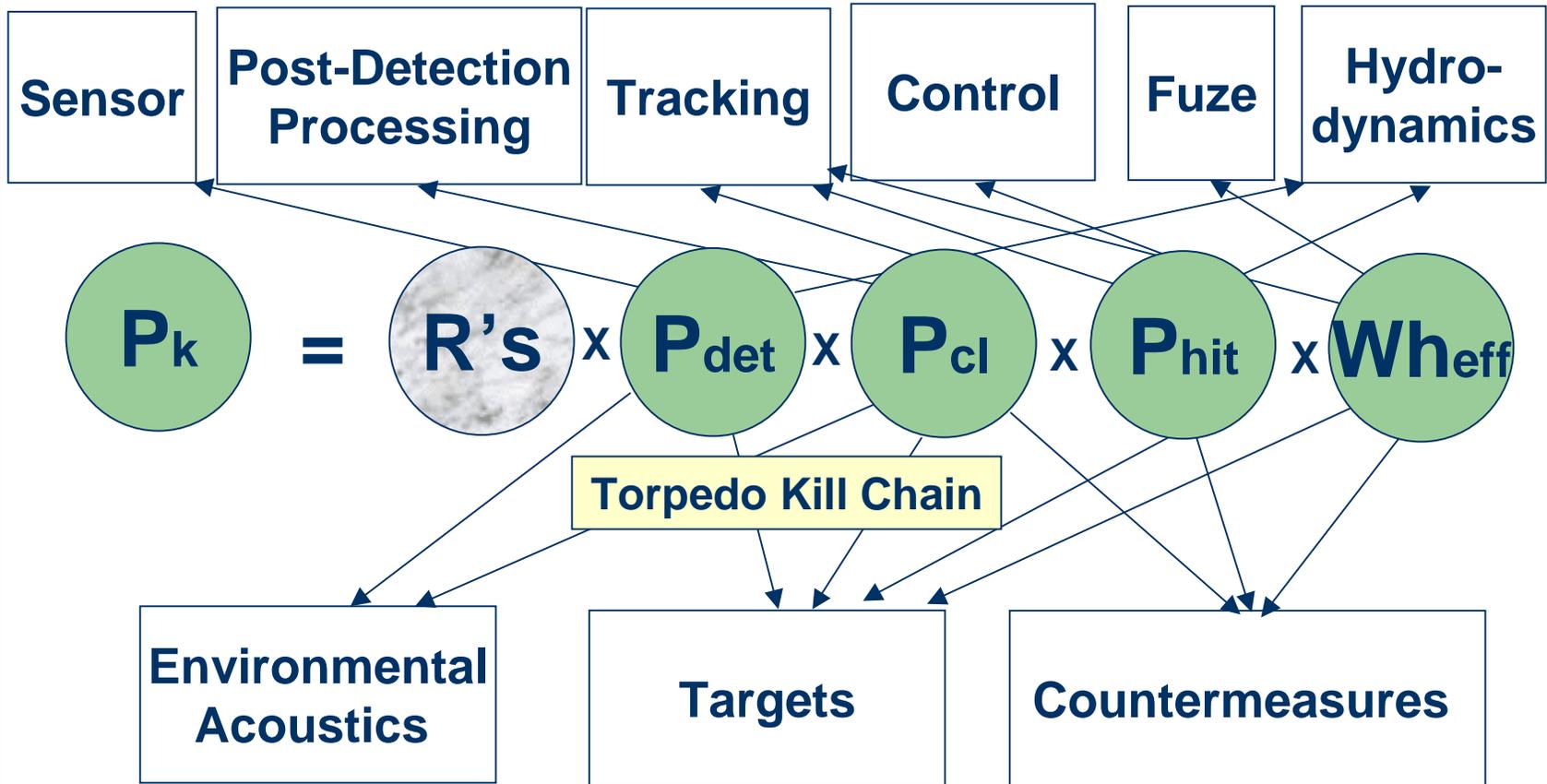
**TEAMS Emphasis:
"Launch-to-Hit"
Analysis**

Military M&S Resolution Levels

Background: High-Level M&S Requirements



Torpedo M&S Components

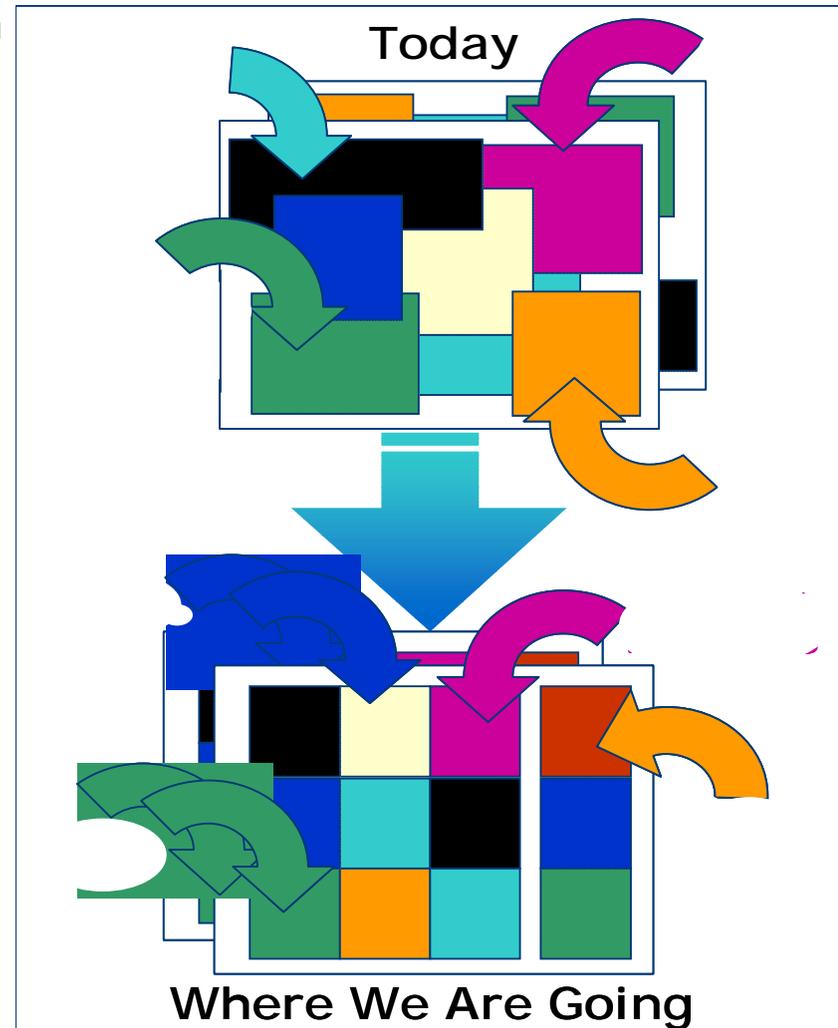


Other "Stimulus" M&S Components



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**



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

- Wants TEAMS as test case for their TOGAF/ Model Driven Architecture (MDA) synergy effort

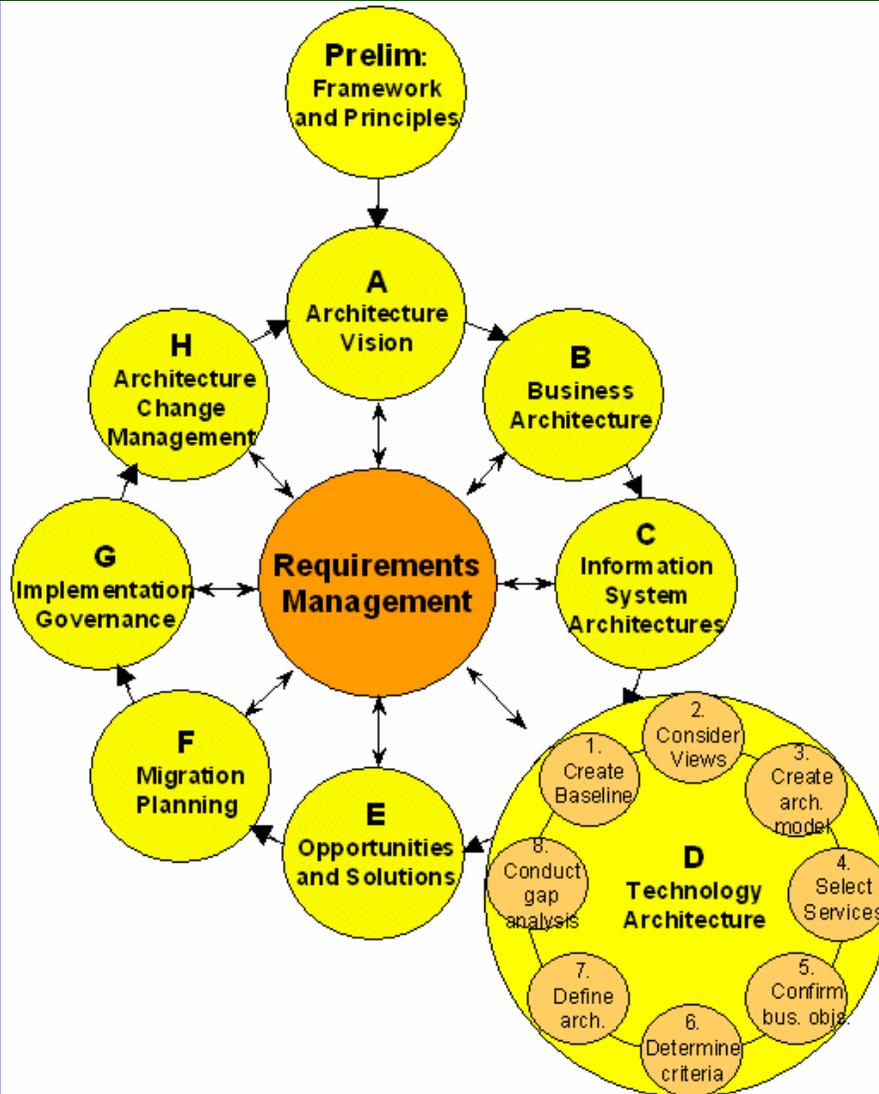


The Open Systems Joint Task Force of the Office of Secretary of Defense (OSD)

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



The Process: TOGAF ADM

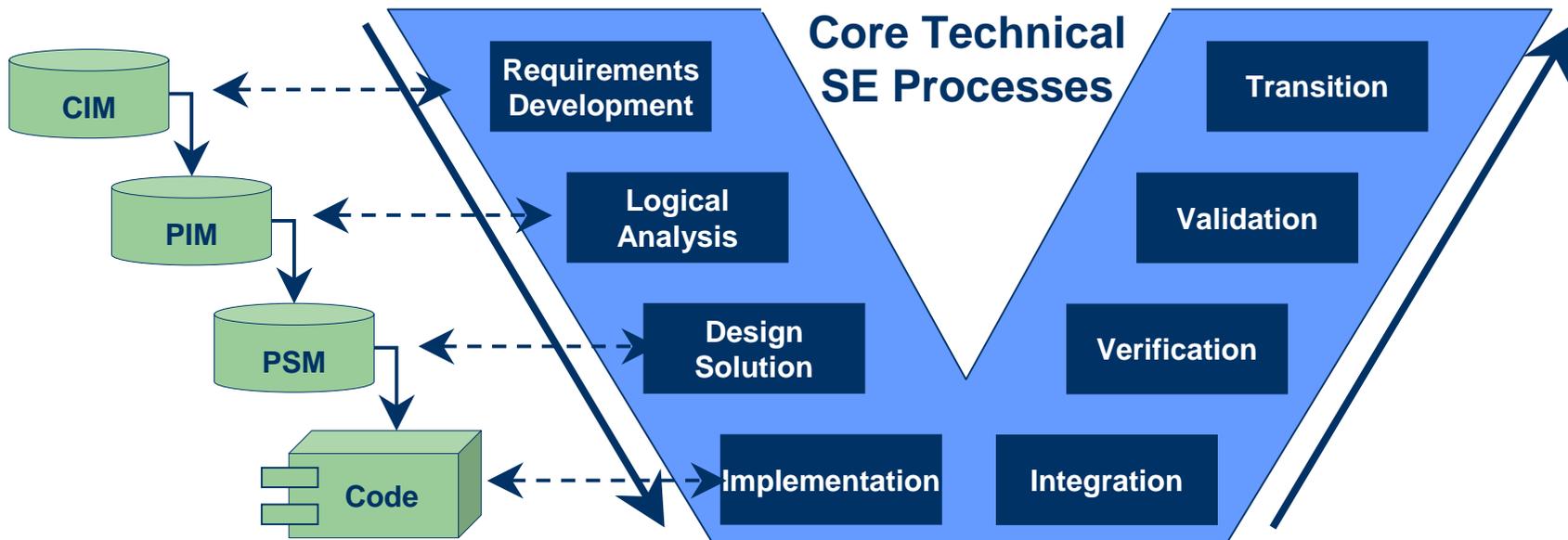


The Open Group:
IT Consortium
Offers Consortia Services

TOGAF:
The Open Group
Architecture Framework

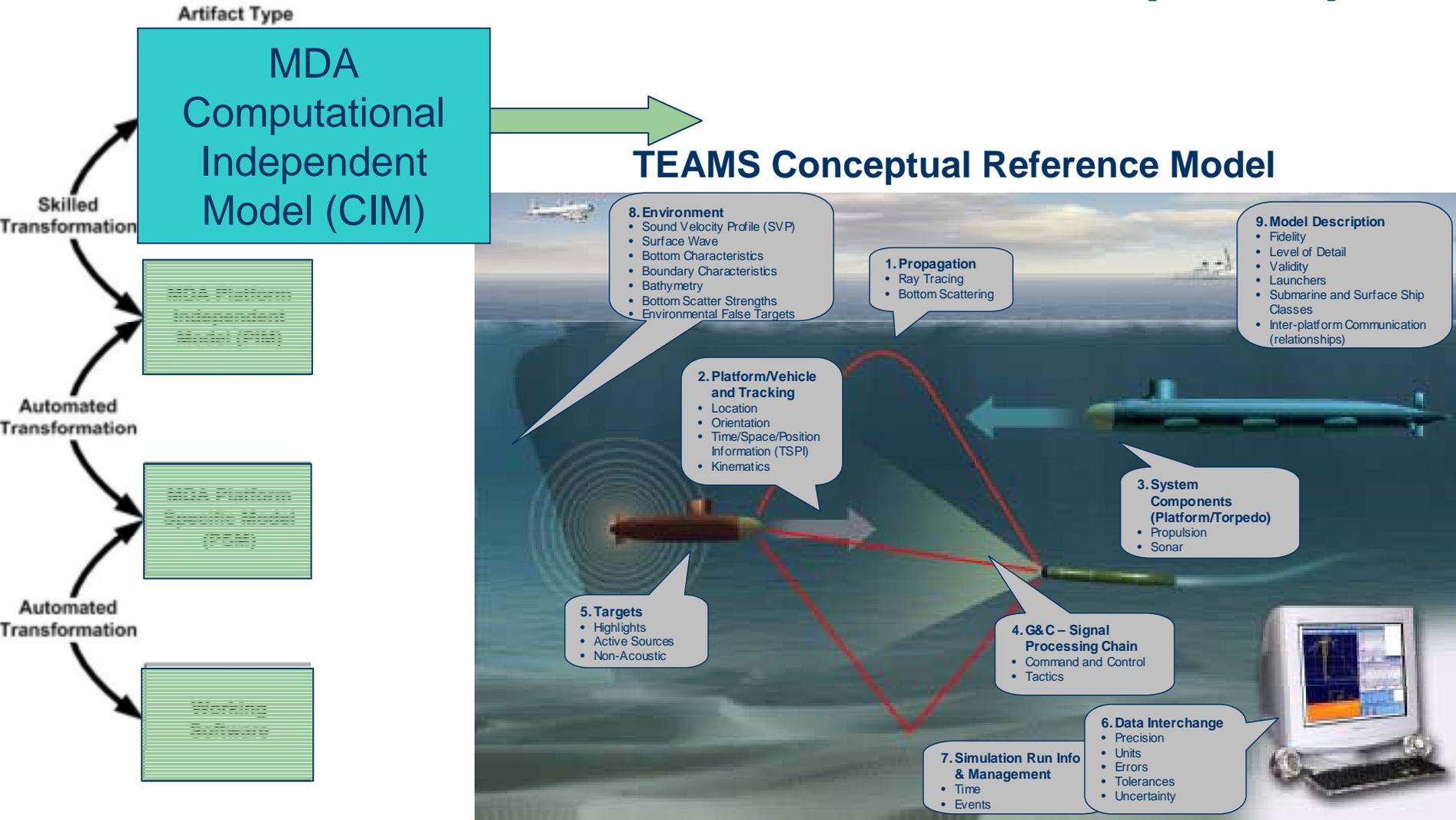
ADM:
Architecture
Development Method

Using MDA in SE Context



The implementation (code) for technology selected by the developer

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)
- Kinematics

3. System Components (Platform/Torpedo)

- Propulsion
- Sonar

5. Targets

- Highlights
- Active Sources
- Non-Acoustic

4. Signal Proc. Chain

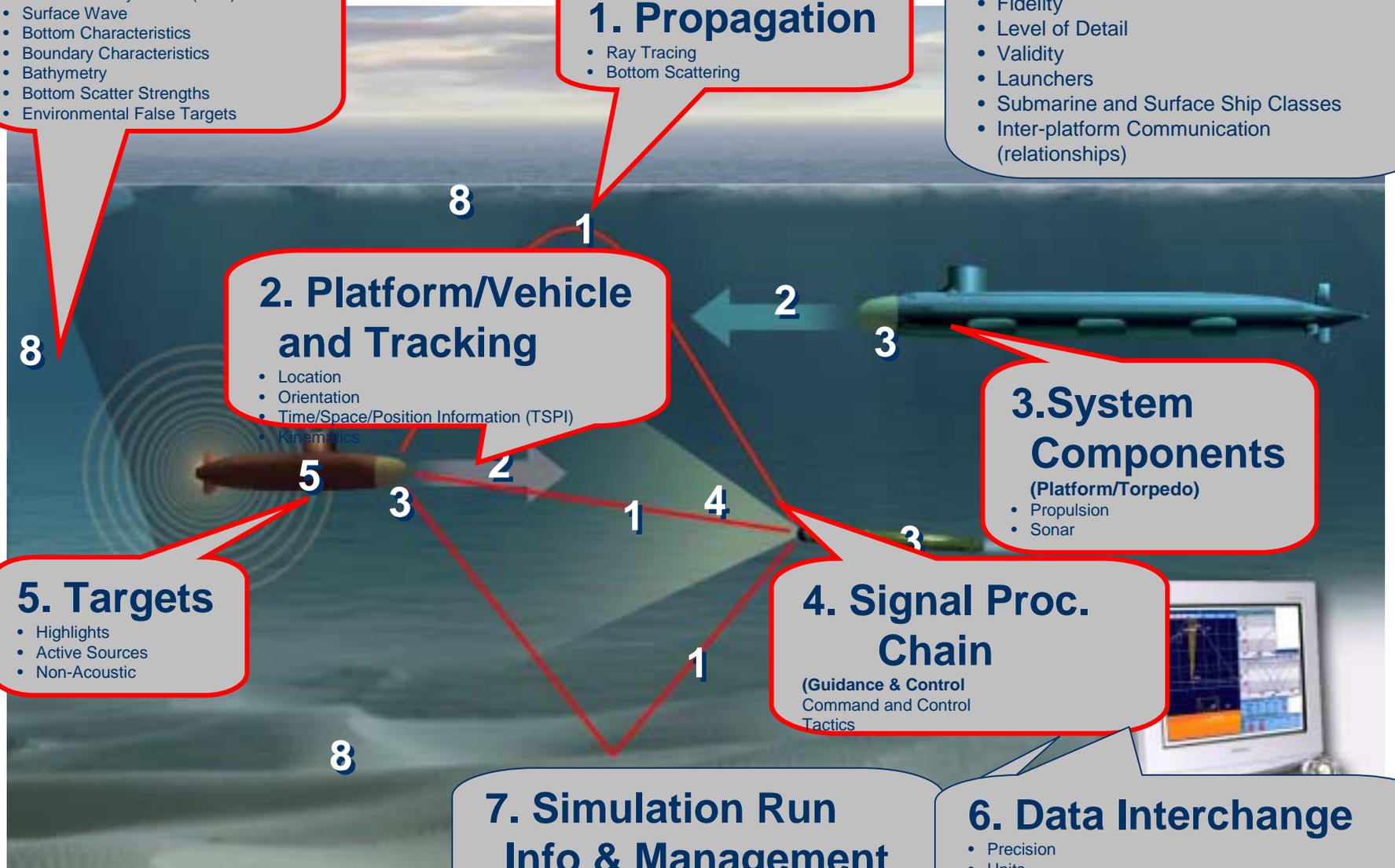
(Guidance & Control
Command and Control
Tactics)

7. Simulation Run Info & Management

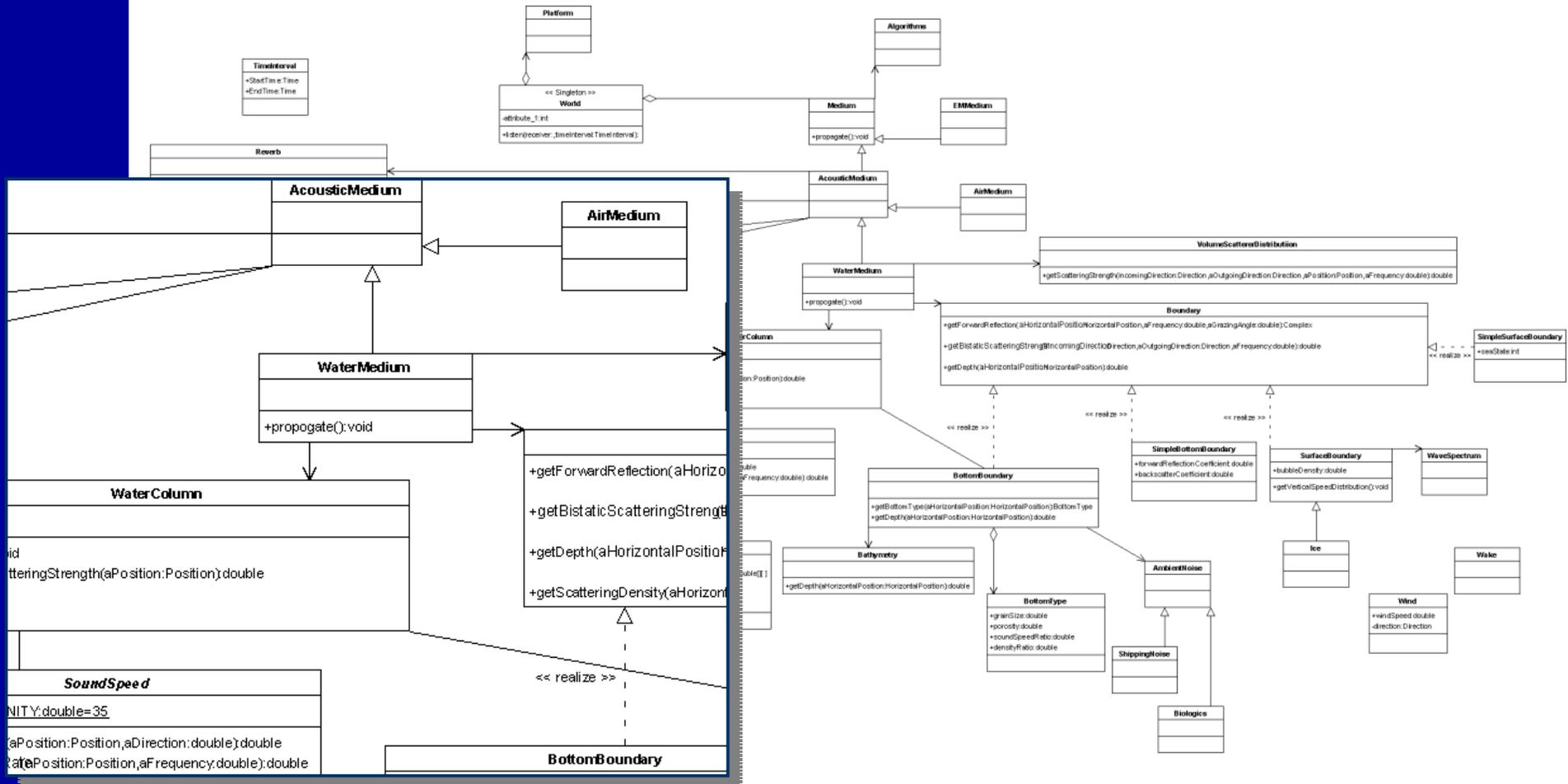
- Time
- Events

6. Data Interchange

- Precision
- Units
- Errors
- Tolerances
- Uncertainty



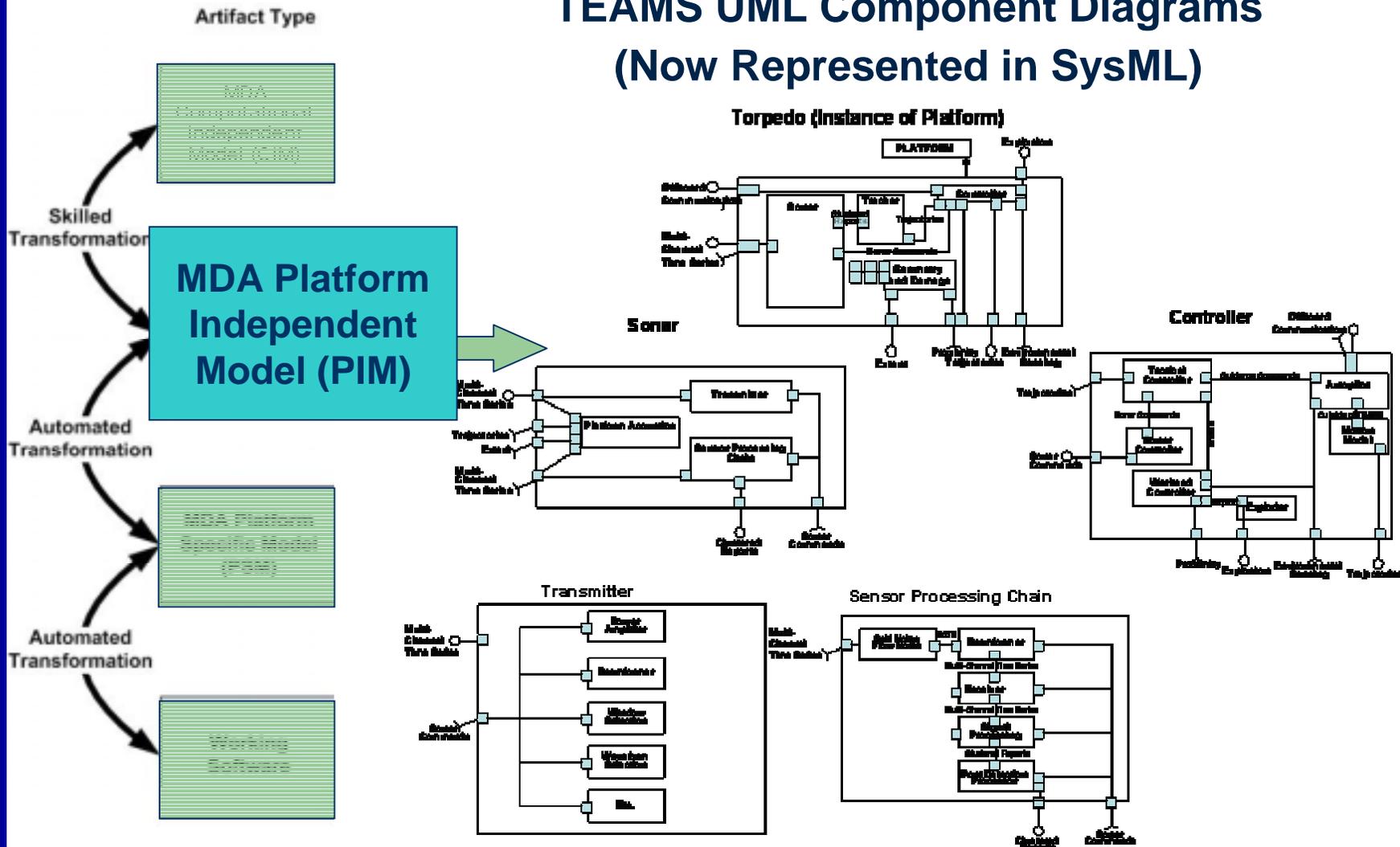
Environment Conceptual Level Diagram



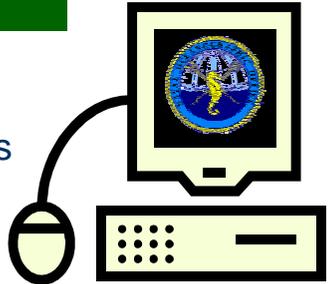
The Method: Model Driven Architecture (MDA)



TEAMS UML Component Diagrams (Now Represented in SysML)



TEAMS PSM: Implementation Planning



In-situ Environmental Data via Web Services

NAVOCEANO
SIPRNET Web Site

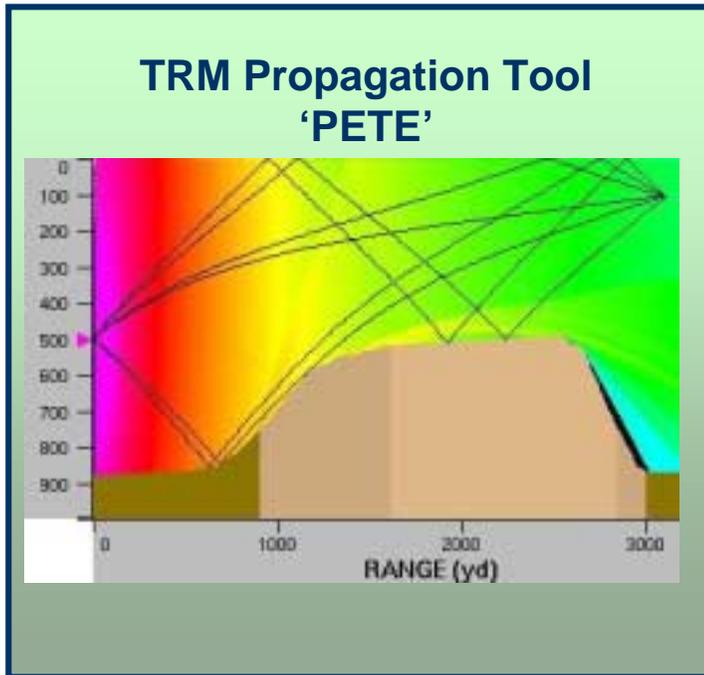


Applied Physics Lab
University of Washington

Fey Rey Propagation Model via HLA*



Defense Modeling and
Simulation Office





TEAMS 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

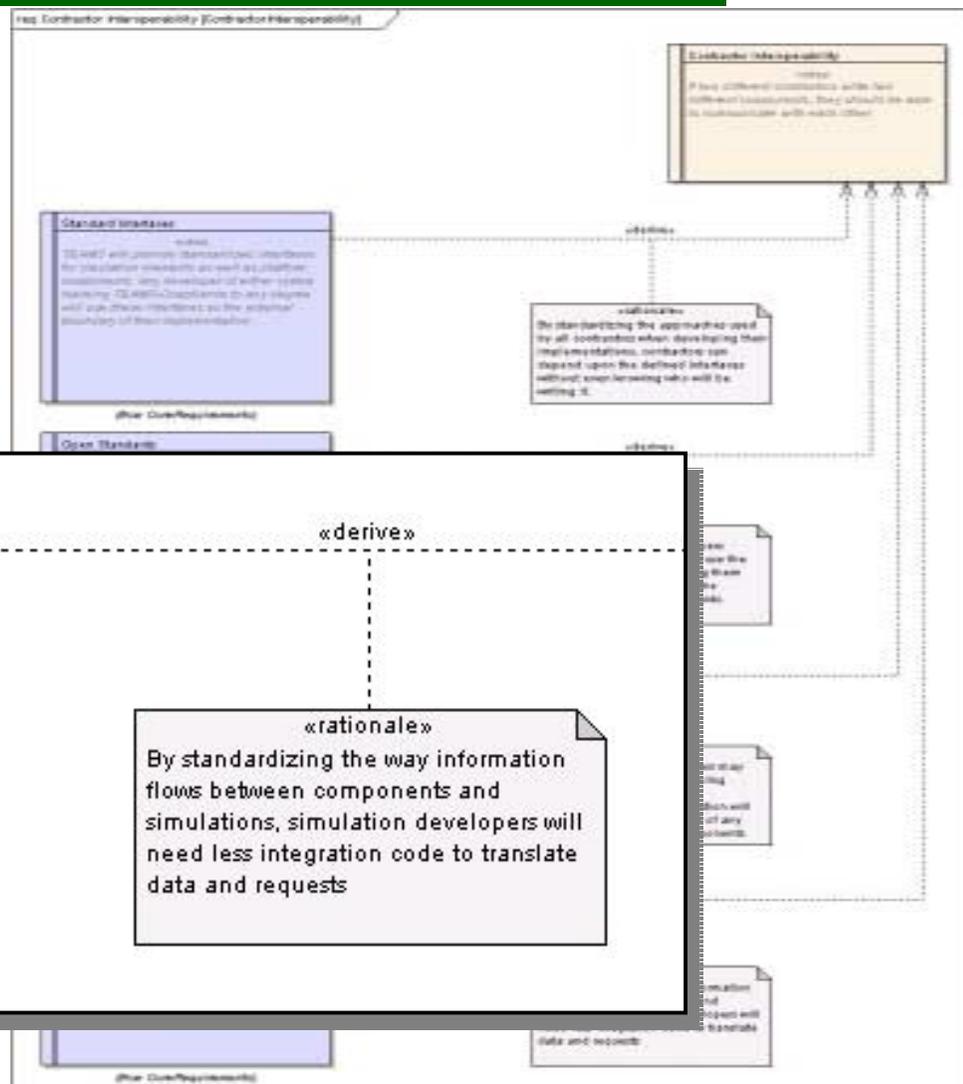
Some mechanism should enable older systems to be pulled into simulations with new interfaces, and newly developed components should have some easily reusable interface to reduce this problem in the future.

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)



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.

(from CoreRequirements)

«derive»

«rationale»

By standardizing the way information flows between components and simulations, simulation developers will need less integration code to translate data and requests

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 Conformance (Conformance)

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 include any platform representation within their design. The interfaces will not be tied to constraints such as language, hardware, operating system, and programming language.

Open Standards

notes
All provide the preferred or standard interfaces. TEAMS will 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 limiting contents of behavior and allow a degree of modifiability between components. These interfaces will be extensible to include new ways of interaction and new extensions of established conventions.

Existing Standards

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

Loosely Coupled Interfaces

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

Tier of Interfaces

notes
The interfaces model will specify interfaces such that higher levels completely replace lower levels. And as interfaces at higher level depend on the lower structure of an interface in its own tier. Disaggregation defined components will carry out within the open interface tier or other communication with the component a parent or child component.

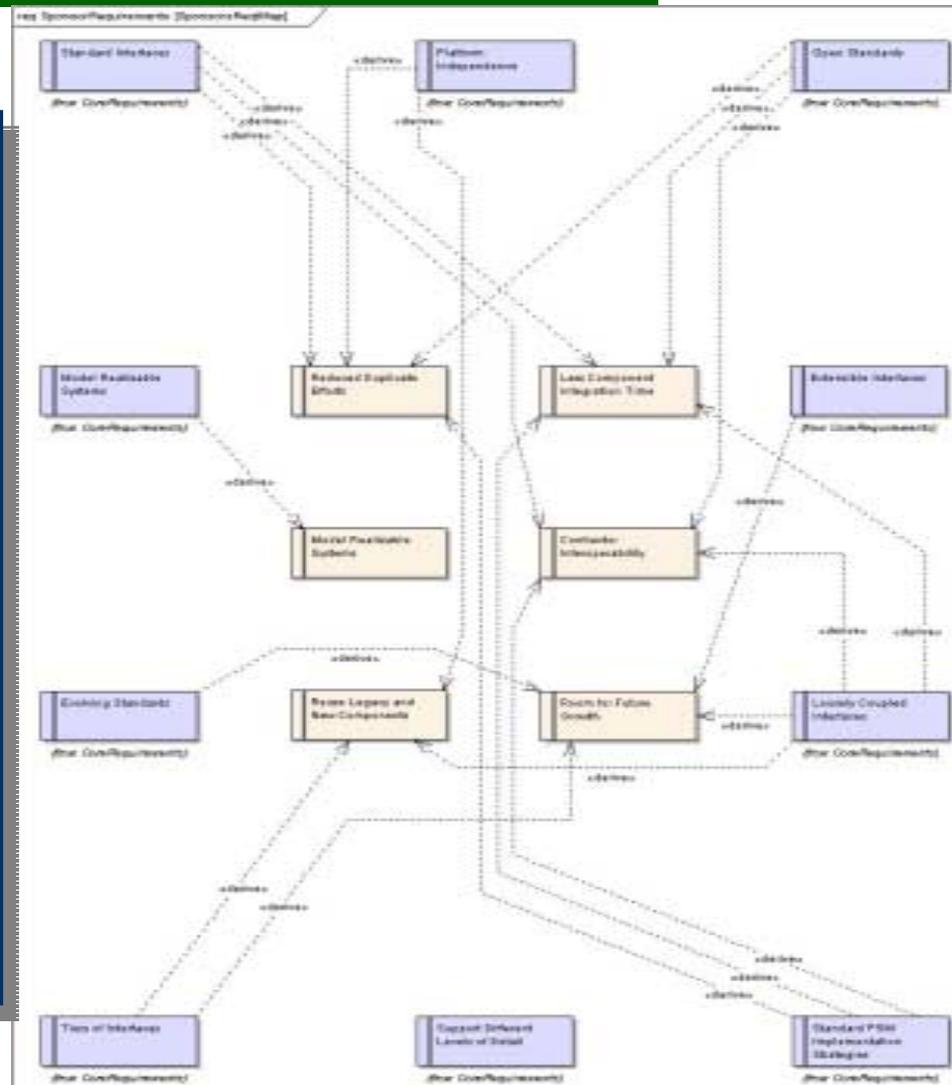
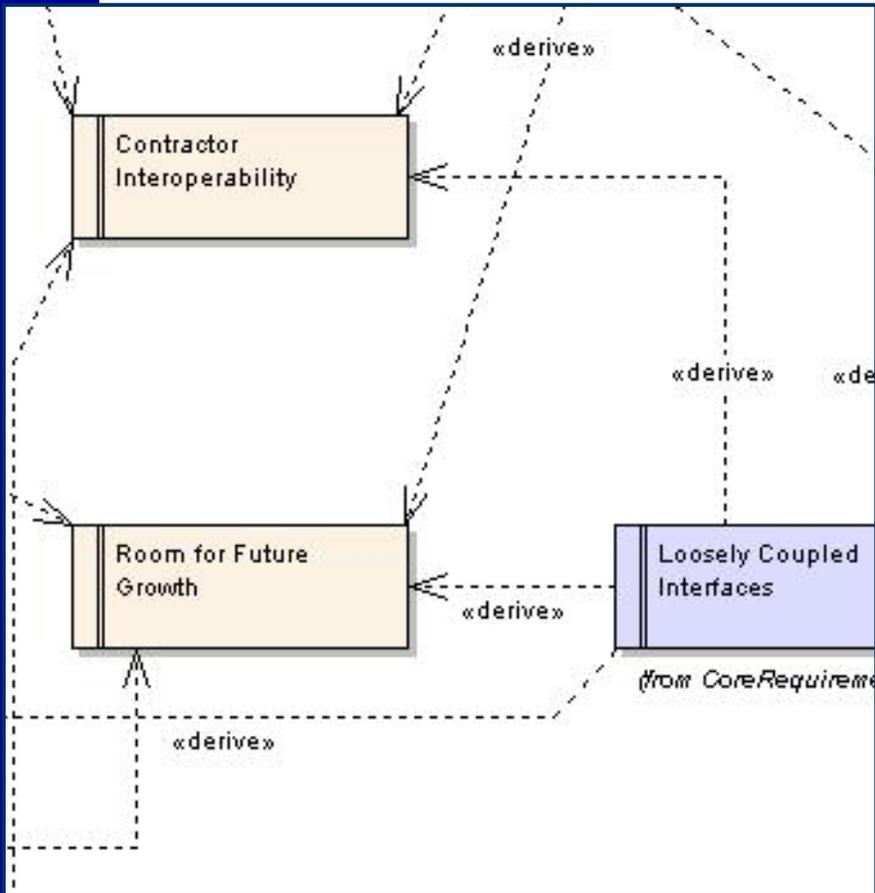
Support Different Levels of Detail

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

Standard PDM Implementation Strategies

notes
To provide interchange of implemented components, TEAMS will provide several explicit implementation strategies. They will present any loss of communication due to a formal strategy that the TEAMS platform implementer will be the primary speech mechanism.

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

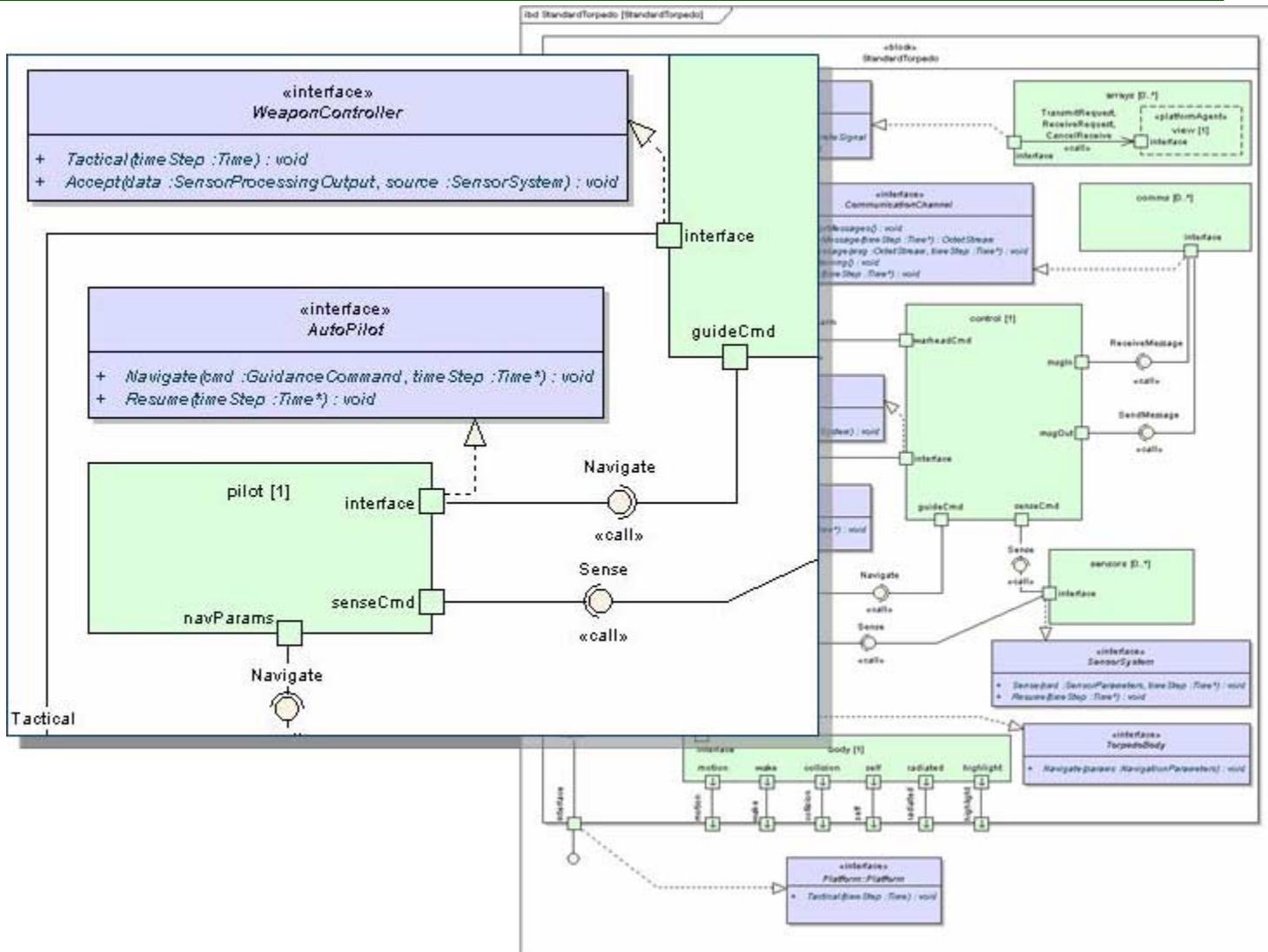
TEAMS Perspective: SysML Pros



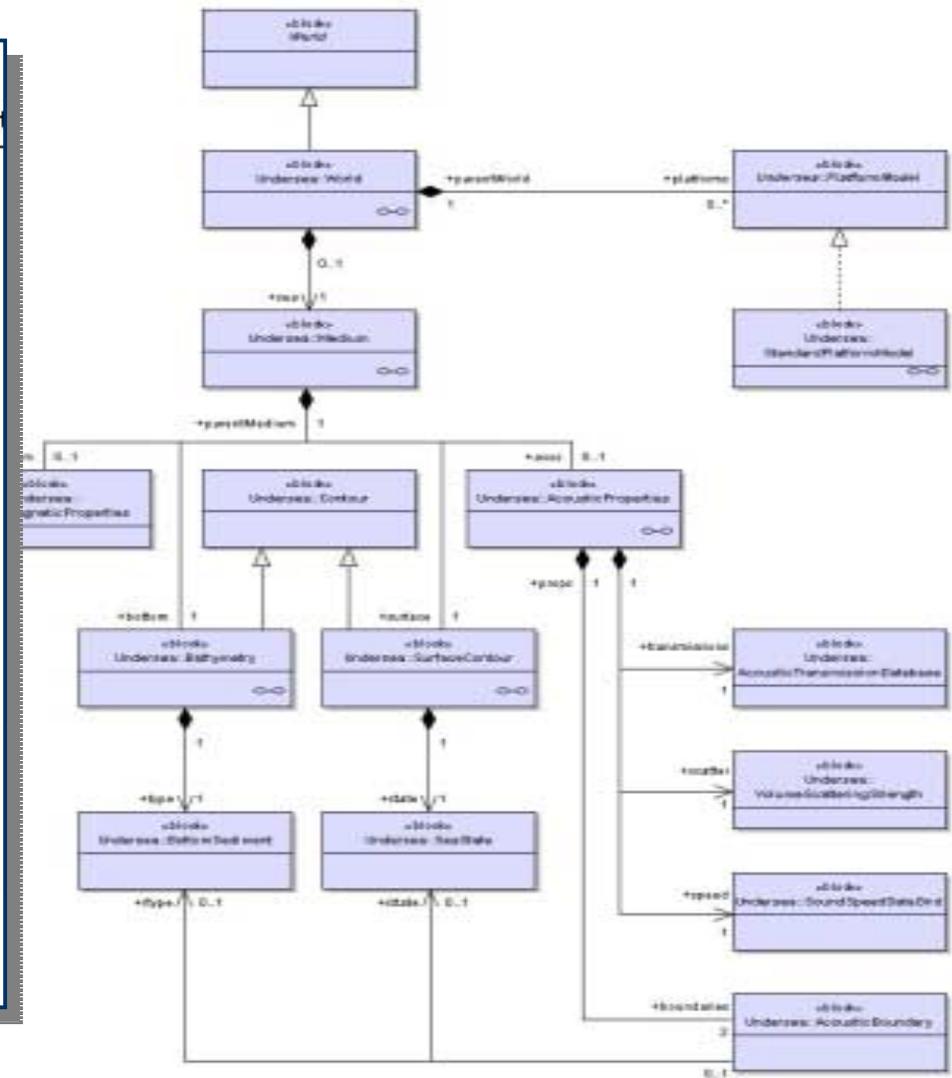
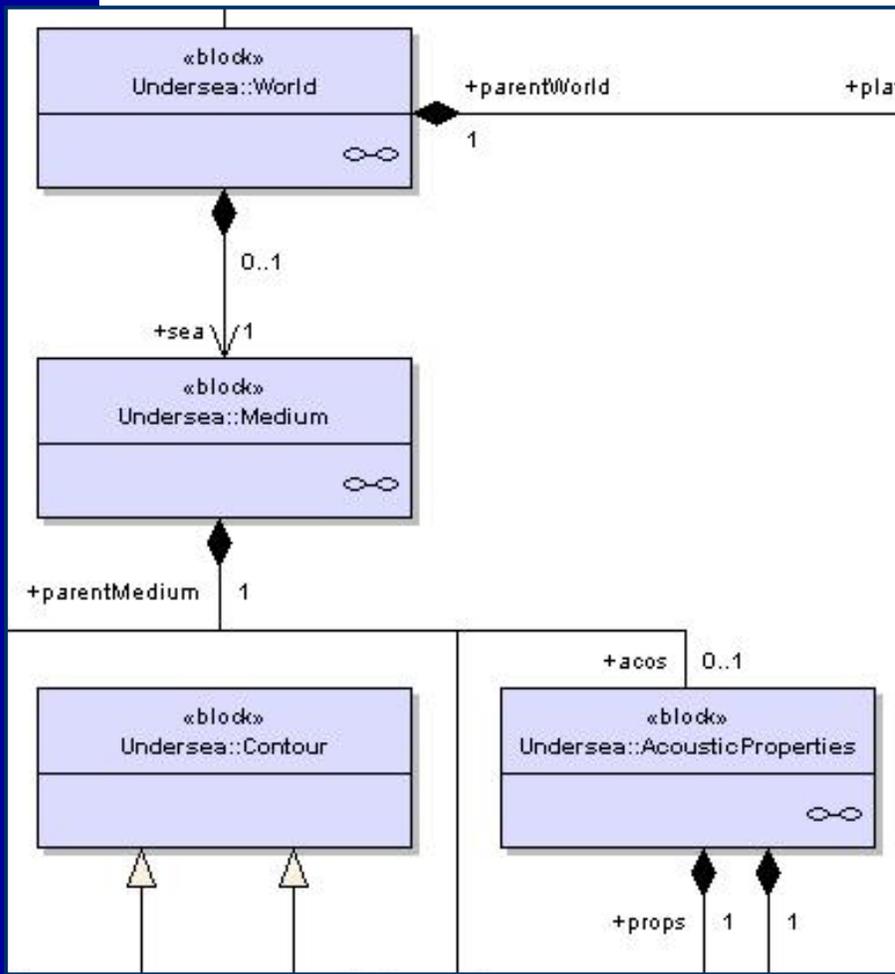
Pros

- **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

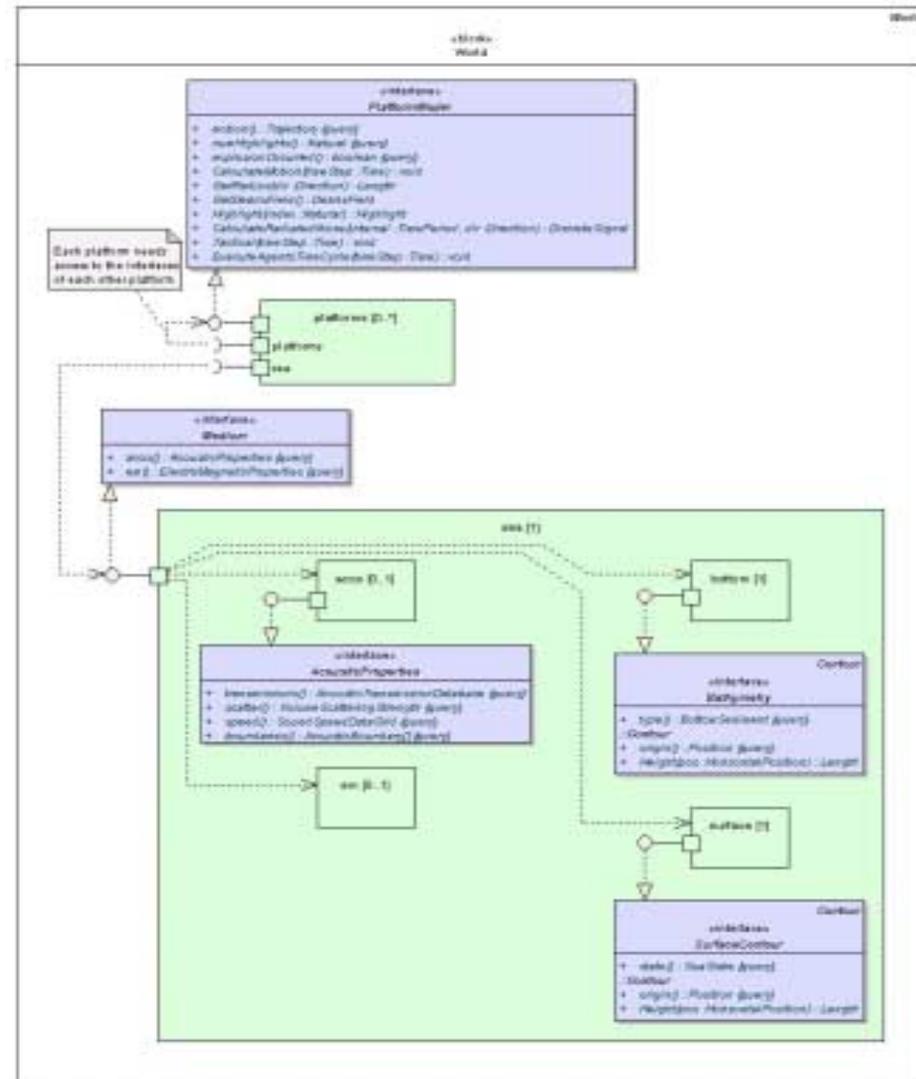
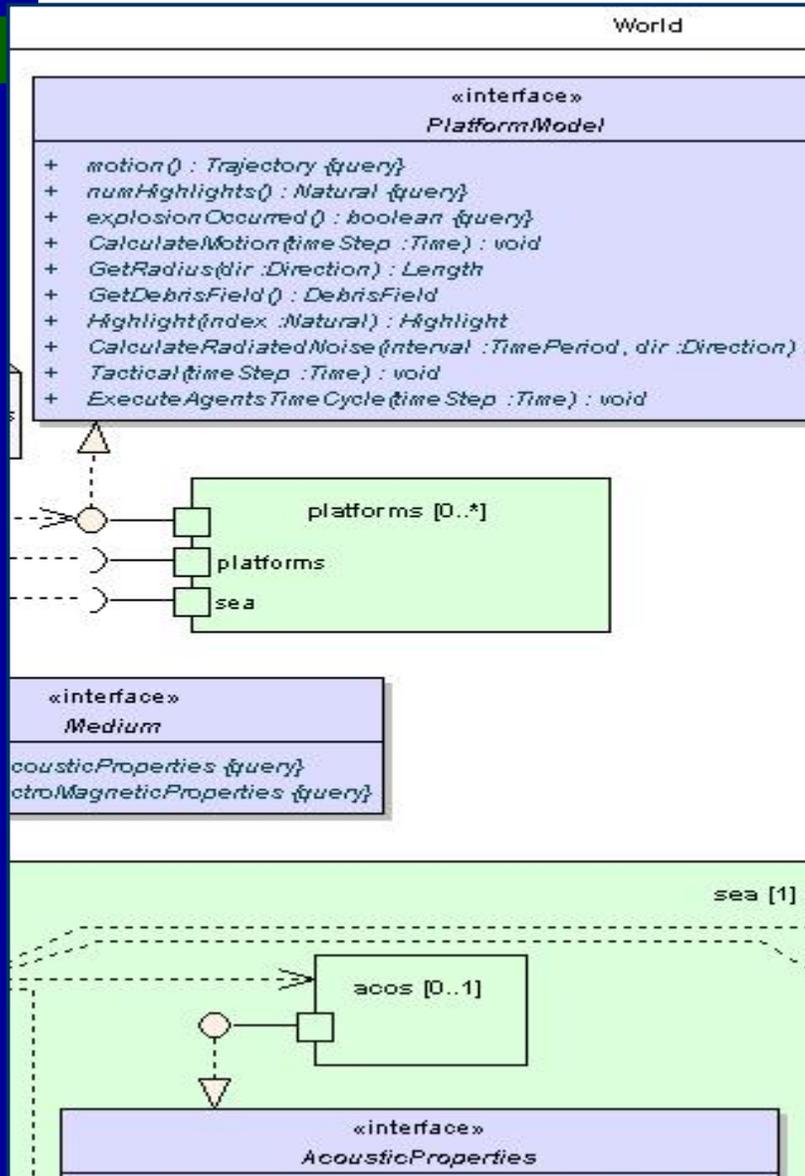
Torpedo Internal Block Definition Diagram



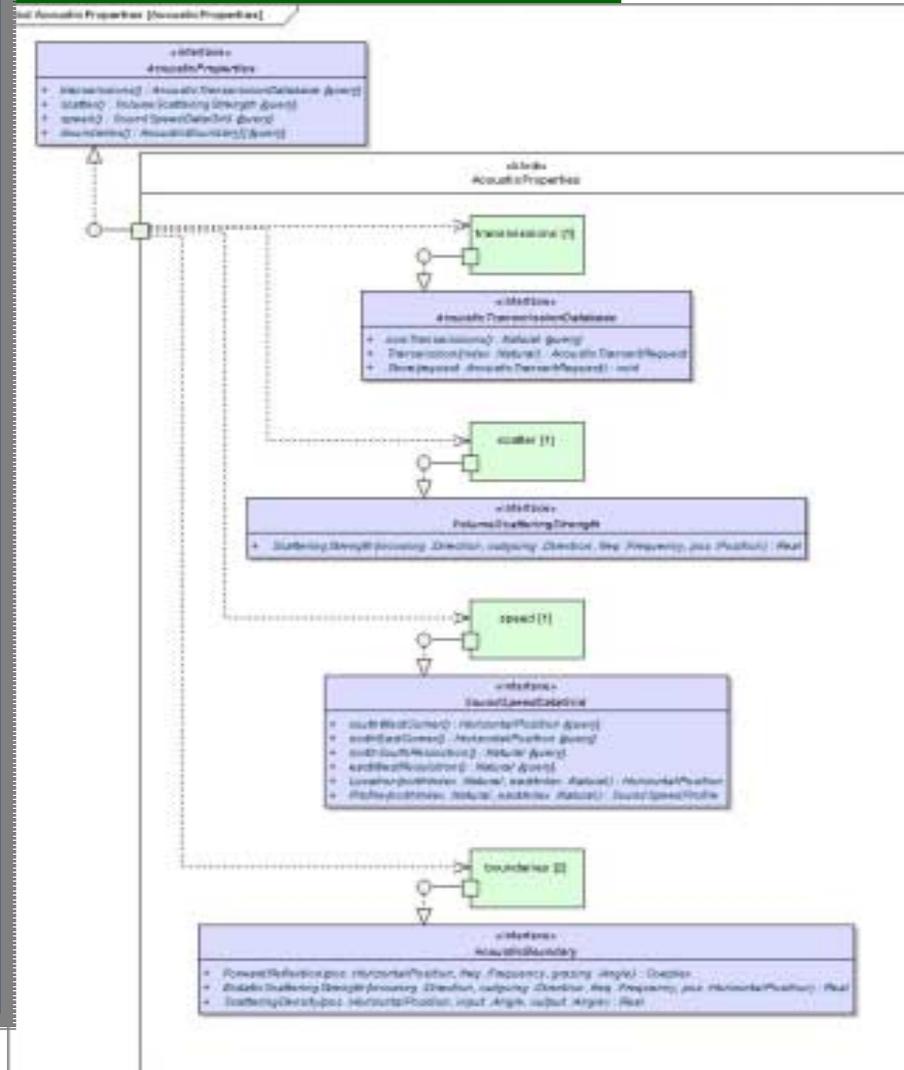
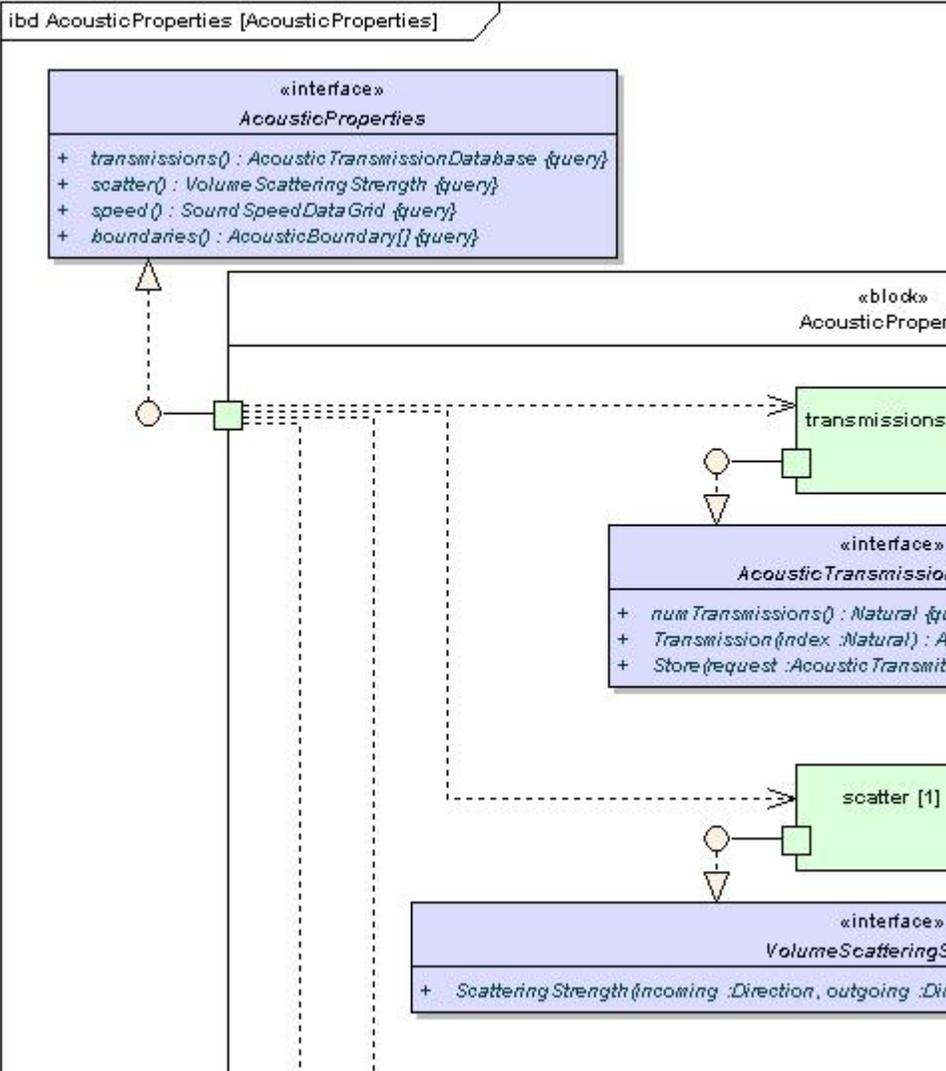
Undersea World Block Definition Diagram



Simulation "World" Internal Block Definition Diagram



Acoustic Properties Internal Block Definition Diagram



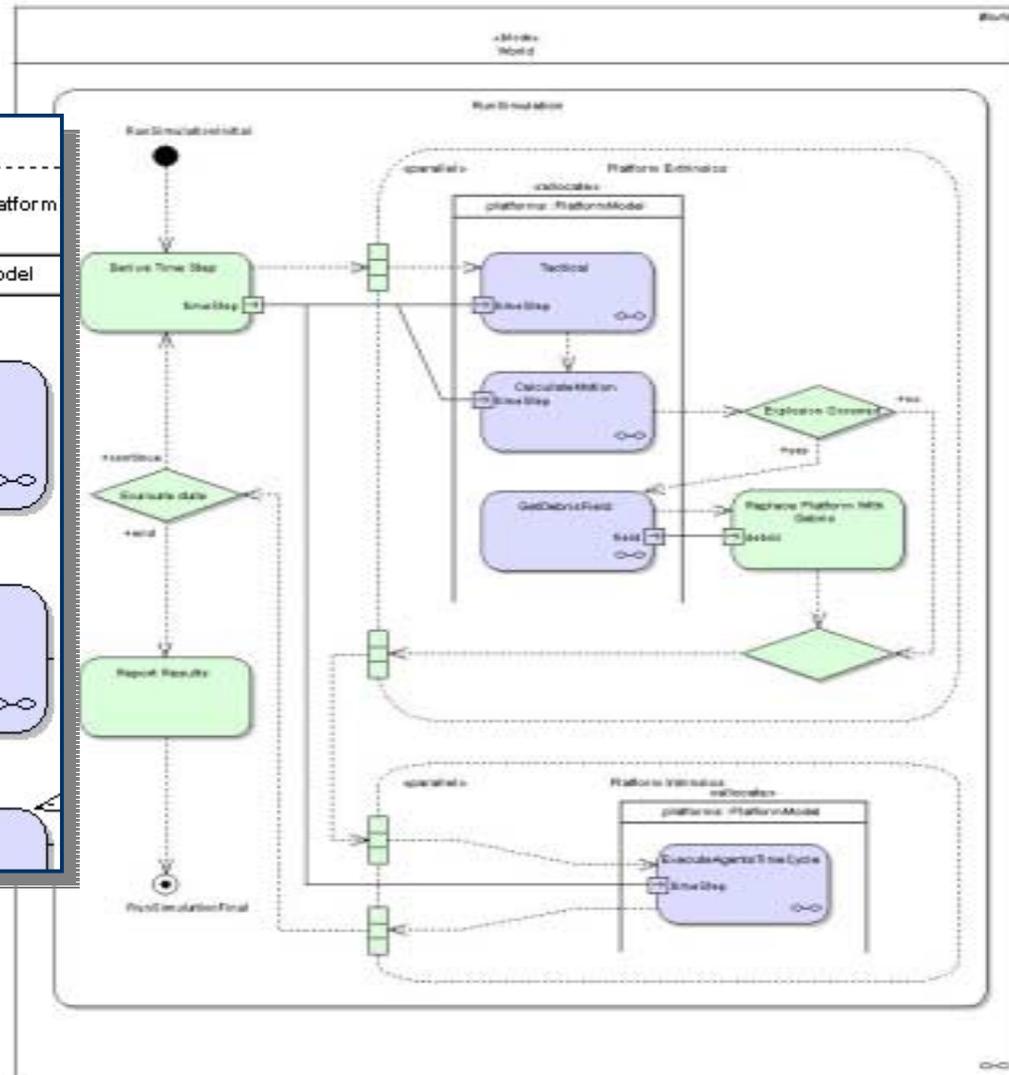
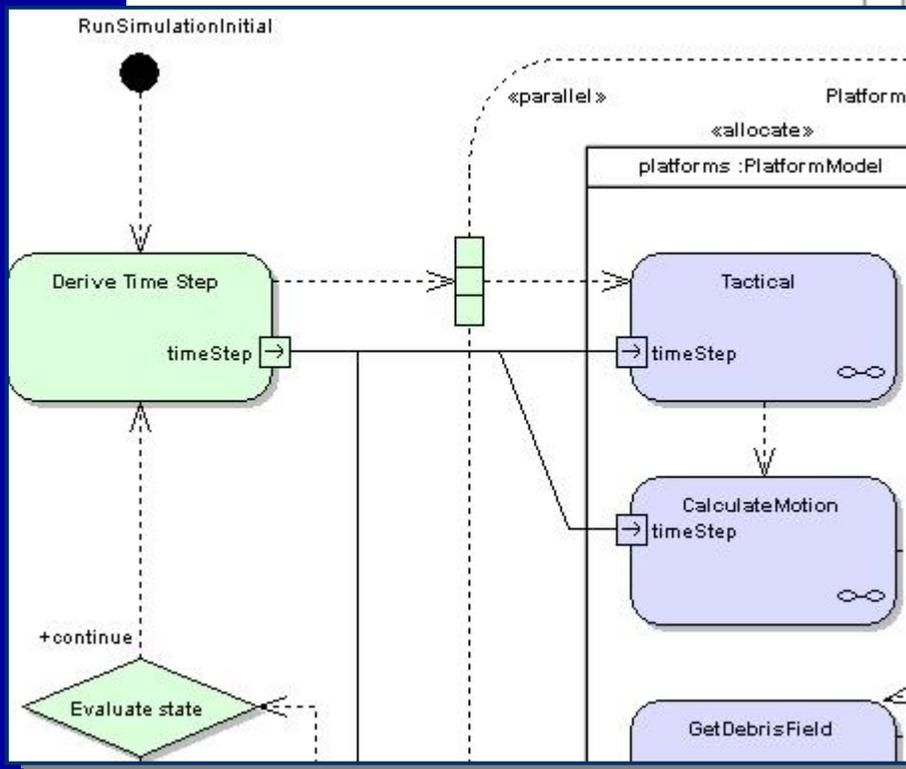
TEAMS Perspective: SysML Pros



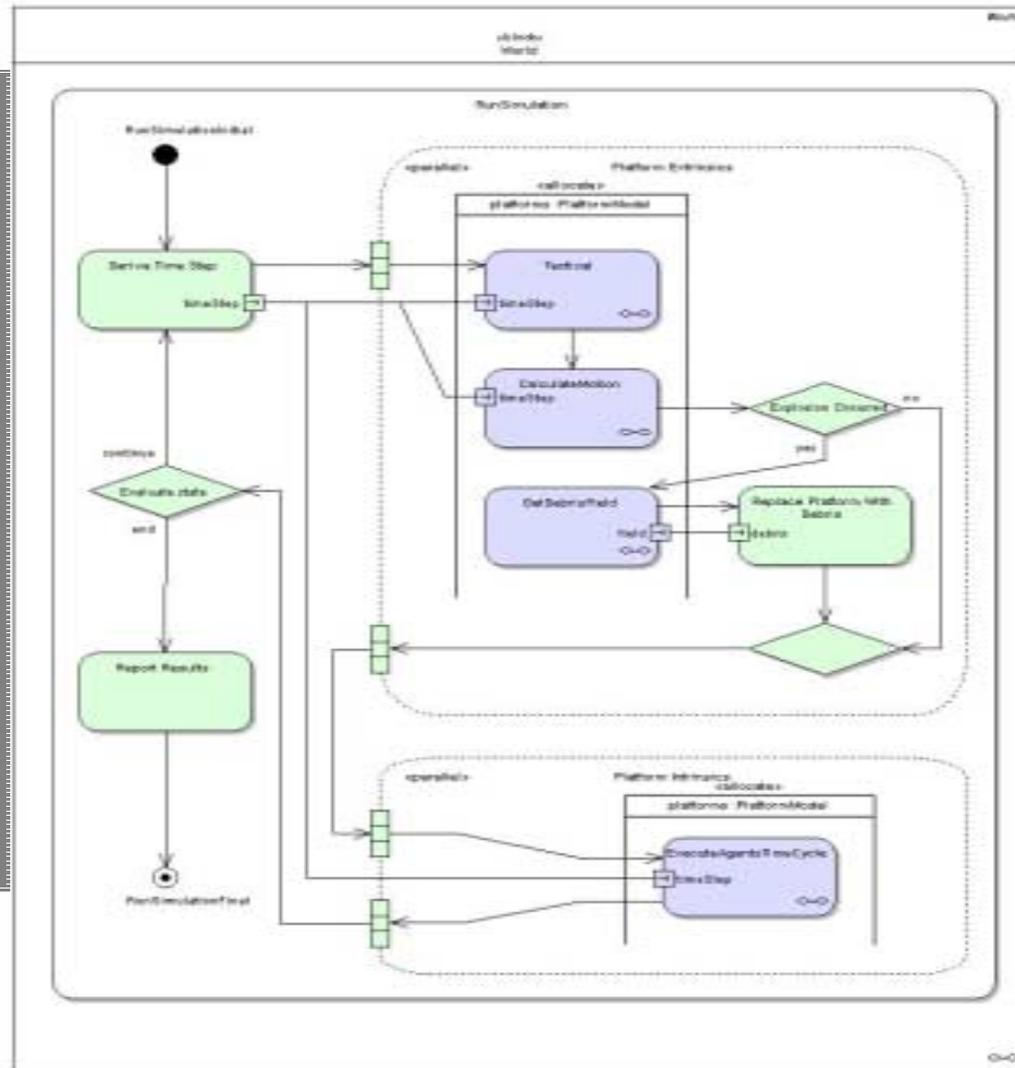
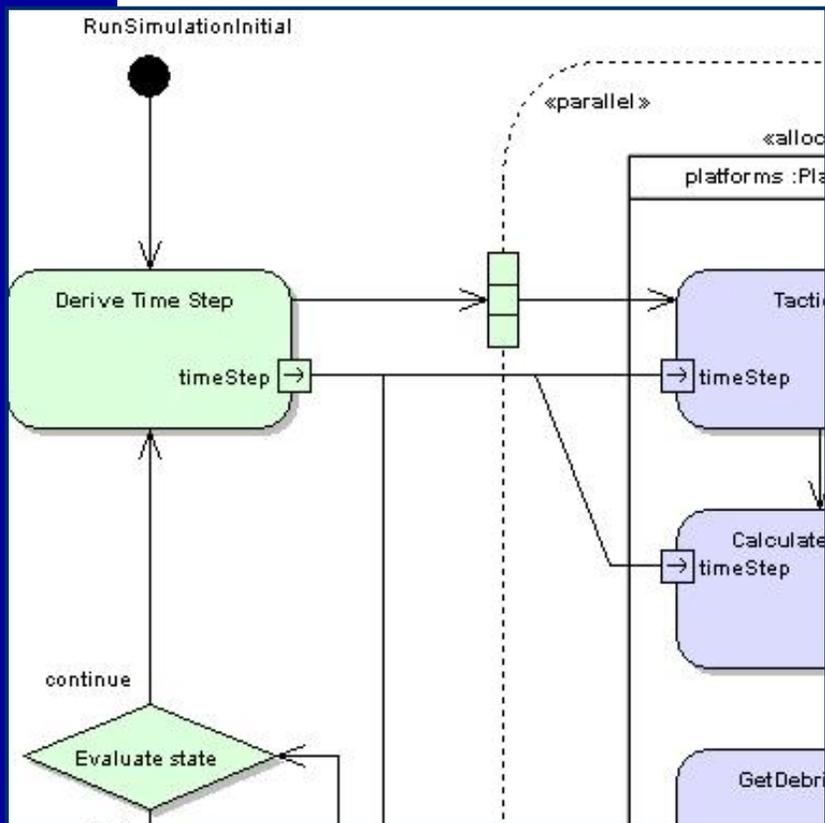
Pros

- **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**

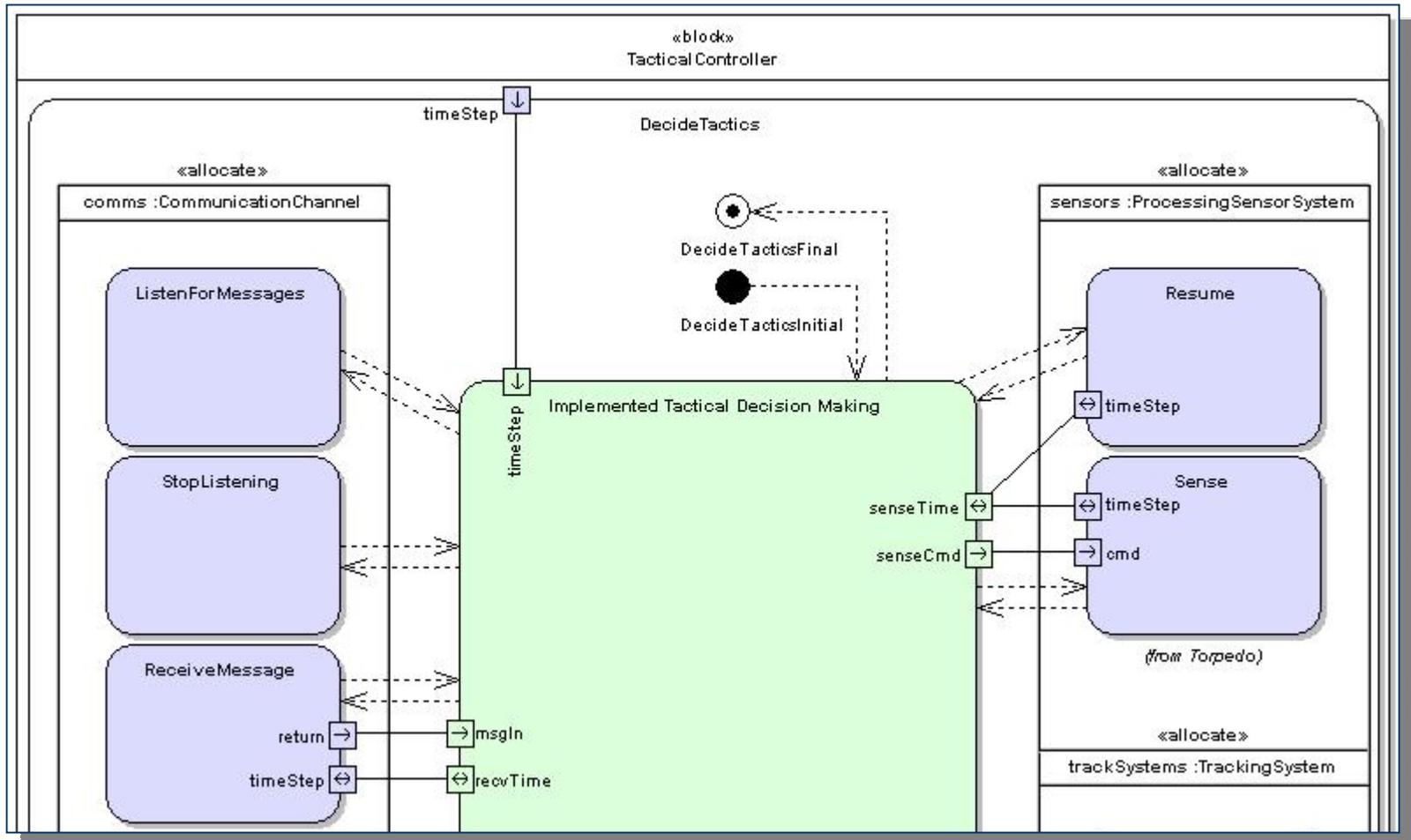
- Exit path dependent on logic within an activity is not accessible and can't be modeled
- Not represented well in either UML or SysML – tactical controller example

- **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
 - 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.”

Domain Model

- Equivalent to the MDA CIM
- Represents the operational domain (e.g., torpedo, submarine platform, targets, and ocean environment)
- Specifies the requirements for the simulation design
- Capture in SysML model
- Parametrics used to specify constraints (e.g., torpedo dynamics, signal propagation)

Simulation Design Model

- Equivalent to the MDA PIM
- Represents the simulation software design
- SysML model "transformed" into simulation model (e.g. Map SysML structure, behavior, and parametrics into simulation components)
- Use SysML allocations to specify the CIM/PIM mapping (i.e., transformation)



Acknowledgements

- **LtCol Telford / Dwayne Hardy and OSD-ATL, for their interest and continued support**
- **David Drumheller and ONR, for their vision and continued support**
- **Sanford Friedenthal, for his expertise and willingness to educate the TEAMS consortium on the nuances of SysML**
- **Members of The Open Group, Object Management Group, and TEAMS Initiative who contributed to the success of SysML Project**
- **Sparx Systems, who provided complimentary licenses for Enterprise Architect 6.5 for this SysML effort**