SysML Overview
Draft Update

SysML Partners
www.sysml.org

OMG SE DSIG Meeting
April 27, 2004

Objectives

- Describe SysML approach for customizing UML 2 to satisfy UML for SE RFP requirements

- Material is “In Process” based on current Draft SysML Specification in preparation for Revised Submission for SysML V1.0 – August 2, 2004

Agenda

- Tuesday, April 27
  - 09:00 - 10:00 Background
  - 10:00 – 10:30 Req'ts and Design Approach
  - 10:30 - 10:45 Break
  - 10:45 - 12:00 Diagram Summary
  - 12:00 - 13:00 Lunch
  - 13:00 - 14:30 Diagram Summary (cont)
  - 14:30 - 15:00 Summary

Background

Motivation

SYSTEMS MODELING LANGUAGE

- Systems Engineers need a standard language for analyzing, specifying, designing, verifying and validating systems
- Many different modeling techniques
  - Behavior diagrams, IDEF0, N2 charts, …
- Lack broad based standard that supports general purpose systems modeling needs
  - satisfies broad set of modeling requirements (behavior, structure, performance, …)
  - integrates with other disciplines (SW, HW, ..)
  - scalable
  - adaptable to different SE domains
  - supported by multiple tools

Why UML for SE ?

SYSTEMS MODELING LANGUAGE

- UML is already de facto standard within software engineering community
- UML is mature and extensible, and can be adapted to support SE requirements
- UML tools and training are widely available
- OMG standardization process supports UML customization for specific domains (e.g., systems engineering)

INCOSE/OMG Joint Initiative

- OMG Systems Engineering Domain Special Interest Group chartered by INCOSE-OMG initiative in 2001
  - create a semantic bridge between ISO 10303-233 standard and ISO/IEC 19501 UML standard
  - create UML extended modeling language for specifying, designing, and verifying complex systems using profiles, or other extensibility mechanisms.
  - provide capability for rigorous transfer of specifications and related information among tools used by systems, software and hardware engineers
  - bridge the semantic gap, the professional engineering discipline gap, and the training gap that exists between systems engineering and software engineering

SE DSIG Tasks

- Drafted UML for SE RFI, issued by OMG in 2002 to validate SE usage and limitations
- Supported development of SE concept model
- Collaborated with UML2 submission teams
- Performed detailed requirements analysis
- Drafted UML for SE Request for Proposal, issued by the OMG in March 2003 (ad/03-03-41)

SysML Partners

- Informal partnership of modeling tool users, vendors, etc.
  - organized in May 2003 to respond to UML for Systems Engineering RFP
- Charter
  - The SysML Partners are collaborating to define a modeling language for systems engineering applications, called Systems Modeling Language™ (SysML™). SysML will customize UML 2 to support the specification, analysis, design, verification and validation of complex systems that may include hardware, software, data, personnel, procedures, and facilities.

SysML Partners (cont.)

- Industry
  - American Systems, Astrium Space, BAE SYSTEMS, Boeing, Deere & Company, Eurostep, Israel Aircraft Industries, Lockheed Martin, Motorola, Northrop Grumman, oose.de, Raytheon, THALES
- Government
  - DoD/OSD, NASA/JPL, NIST
- Tool Vendors
  - Artisan, Ceira, Gentleware, IBM/Rational, I-Logix, PivotPoint Technology, Popkin, Project Technology, 3SL, Telelogic, Vitech
- Liaisons
  - AP-233, CCSDS, EAST, INCOSE, Rosetta

SysML Milestones

- UML for SE RFP issued – March 28, 2003
- Kickoff meeting – May 6, 2003
- Overview presentation to OMG ADTF – Oct 27, 2003
- Initial draft submitted to OMG – Jan 12, 2004
- INCOSE Review – January 25-26, 2004
- INCOSE Review – May 25, 2004
- Final draft submitted to OMG – Aug 2 (goal)
- OMG technology adoption – Q4 2004

Internal Process

- Applying systematic approach to language development
  - requirements analysis
  - language architecture & design
  - verification & validation
  - requirements traceability
  - reviews with stakeholders
- Partnership collaboration mechanisms
  - weekly telecons
  - monthly physical meetings
  - intranet, web site, and mailing lists

SYSTEMS MODELING LANGUAGE

Requirements Review

UML for SE Request For Proposal

- Specifies requirements for SE modeling language
- Joint requirements reviewed by OMG/INCOSE/AP-233
- Issued by OMG on March 28, 2003
  - OMG Doc# ad/03-03-41
  - http://syseng.omg.org/UML_for_SE_RFP.htm

Scope of RFP

- Focuses on general purpose system modeling
  - physical systems including software and hardware intensive systems
  - system-level vs. hw/sw implementation models (code, 3D geometry, VHDL, ...)
  - integration with discipline specific models (i.e., reliability, safety, ...)

Requirements Summary

- Structure
  - e.g., system hierarchy, interconnection
- Behavior
  - e.g., function-based behavior, state-based behavior
- Properties
  - e.g., parametric models, time property
- Requirements
  - e.g., requirements hierarchy, traceability
- Verification
  - e.g., test cases, verification results
- Other
  - e.g., trade studies, spatial relationships

Evaluation Criteria

- Ease of use
- Unambiguous
- Precise
- Complete
- Scalable
- Adaptable to different domains
- Capable of complete model interchange
- Evolvable
- Process and method independent
- Compliant with UML metamodel
- Verifiable

Requirements Traceability

Requirement #	Requirement name			SysML Diagram
		Planned for V1.0	Planned for V1.X	
6.5	Mandatory Requirements			
6.5.1	Structure	Y		Structure Diagrams
6.5.1.1	System hierarchy	Y		Class, Assembly
6.5.1.2	Environment	Y		Class, Assembly
6.5.1.3	System interconnection	Y		Assembly
6.5.1.3.1	Port	Y		Assembly
6.5.1.3.2	System boundary	Y		Assembly
6.5.1.3.3	Connection	Y		Assembly
6.5.1.4	Deployment of components to nodes	Y		Assembly
6.5.2	Behavior	Y		Behavior Diagrams
6.5.2.1	Functional Transformation of Inputs to	Y		Activity
6.5.2.1.1	Input/Output	Y		Activity, Assembly
6.5.2.1.2	System store	Y		Assembly
6.5.2.1.3	Function	Y		Activity
6.5.2.2	Function activation/deactivation	Y		Activity, Sequence, State
6.5.2.2.1	Control input	Y		Activity
6.5.2.2.2	Control operator	Y		Activity
6.5.2.2.3	Events and conditions	Y		Activity, Sequence, State
6.5.2.3	Function-based behavior	Y		Activity, Sequence
6.5.2.4	State-based behavior	Y		State
6.5.2.4.1	Activation time	Y		Timing
6.5.2.5	Allocation of behavior to systems	Y		Activity

Requirements Traceability (cont.)

SYSTEMS MODELING LANGUAGE

Requirement #	Requirement name			SysML Diagram
		Planned for V1.0	Planned for V1.X	
6.5.3	Property	Y		Parametric
6.5.3.1	Property type	Y		Auxilliary
6.5.3.2	Property value	Y		Class
6.5.3.3	Property association	Y		Parametric
6.5.3.4	Time property	Y		Parametric
6.5.3.5	Parametric model	Y		Parametric
6.5.3.6	Probe	N		Port on Assembly
6.5.4	Requirement	Y		Requirement
6.5.4.1	Requirement specification	Y		Requirement
6.5.4.2	Requirement properties	Y		Requirement
6.5.4.3	Requirement relationships	Y		Requirement
6.5.4.4	Problem		Y	Causal analysis (Logic)
6.5.4.5	Problem association		Y	Causal analysis (Logic)
6.5.4.6	Problem cause		Y	Causal analysis (Logic)
6.5.5	Verification		Y	Verification
6.5.5.1	Verification Process		Y	Verification
6.5.5.2	Test case	Y		Requirement, Verification
6.5.5.3	Verification result		Y	Verification
6.5.5.4	Requirement verification		Y	Verification
6.5.5.5	Verification procedure		Y	Verification
6.5.5.6	Verification system		Y	Verification
6.5.6	Other			
6.5.6.1	General relationships	Y		Class
6.5.6.2	Model views	Y		Auxilliary
6.5.6.3	Diagram types	Y		All Diagrams

Requirements Traceability (cont.)

Requirement #	Requirement name			SysML Diagram
		Planned for V1.0	Planned for V1.X	
6.6	Optional Requirements			
6.6..1	Topology		Y ?	N/A
6.6..2	Documentation	Y		Diagram Chapter
6.6..3	Trade-off studies and analysis		Y	Parametrics, Decision Tree
6.6..4	Spatial representation		Y	
6.6.4.1	Spatial reference		Y	
6.6.4.2	Geometric relationships		Y	
6.6..5	Dynamic structure		Y	
6.6..6	Executable semantics	Partial	Y	Activity
6.6..7	Other behavior modeling paradigms		?	
6.6..8	Integration with domain-specific models	Partial	Y	AP-233 Alignment
6.6..9	Testing Model		Y	Testing Profile

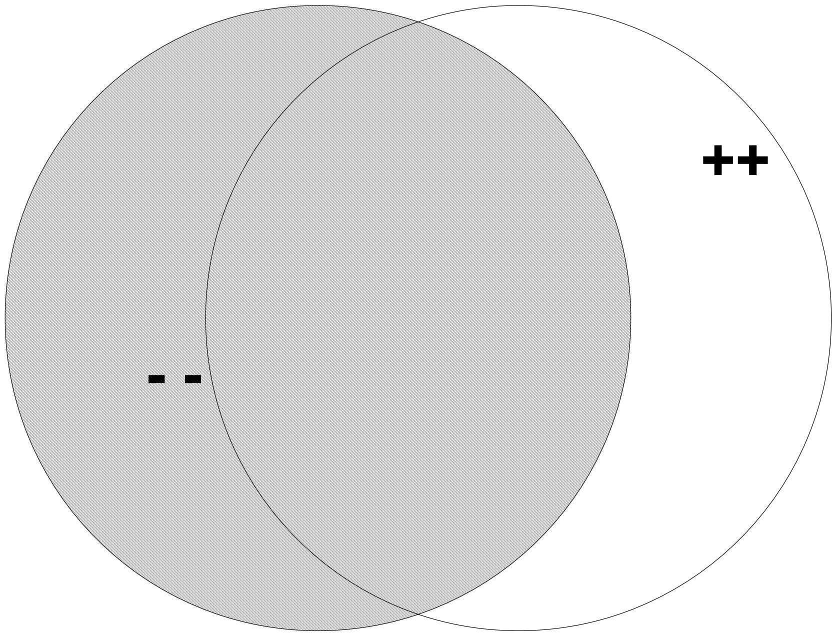
SYSTEMS MODELING LANGUAGE

Design Approach

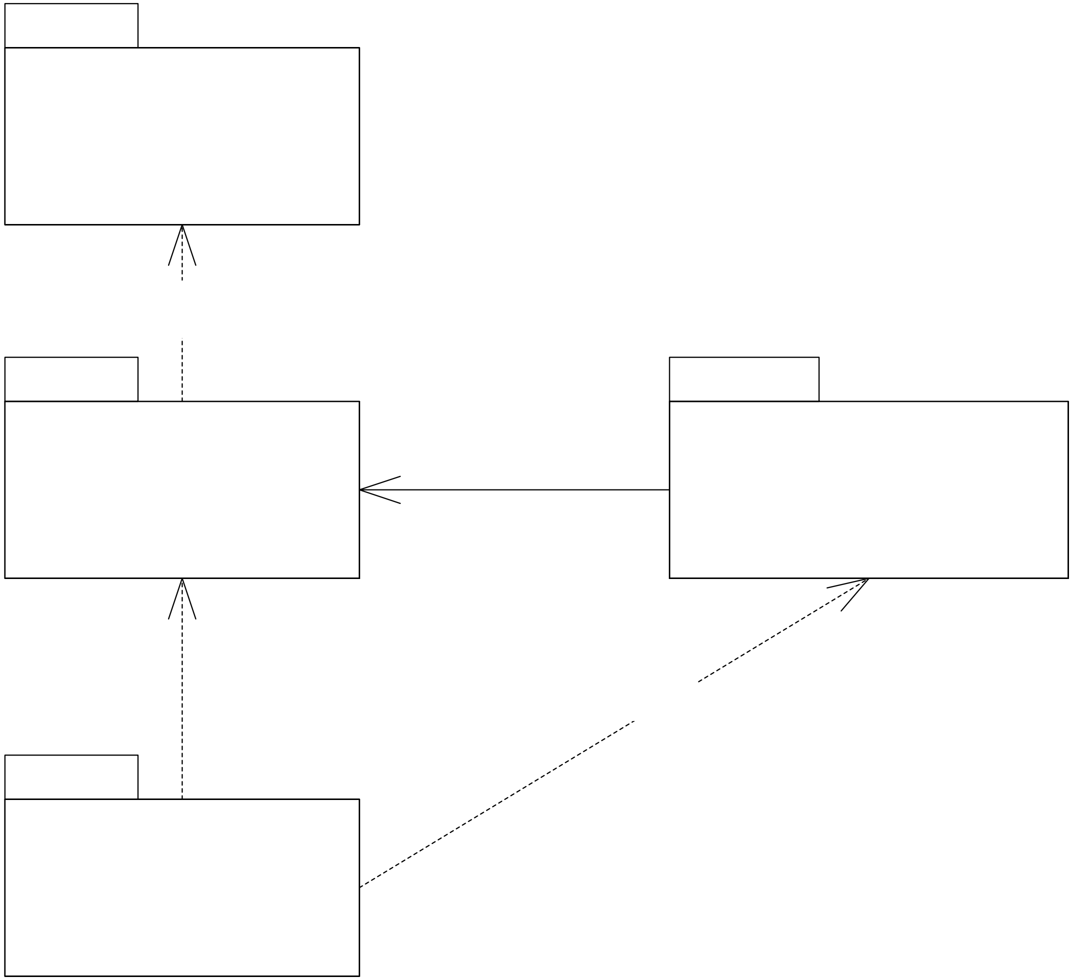
Design Principles

- Reuse and extension
  - select the subset of UML 2.0 that is reusable for SE applications
  - add new constructs and diagrams needed for SE
  - UML2^{++--}
- Incremental development
  - extend the language incrementally, using SE feedback to ensure new extensions are valid
  - prevent scope and schedule creep
- Architectural alignment
  - align with evolving AP-233 SE Data Interchange Standard

UML 2++/--

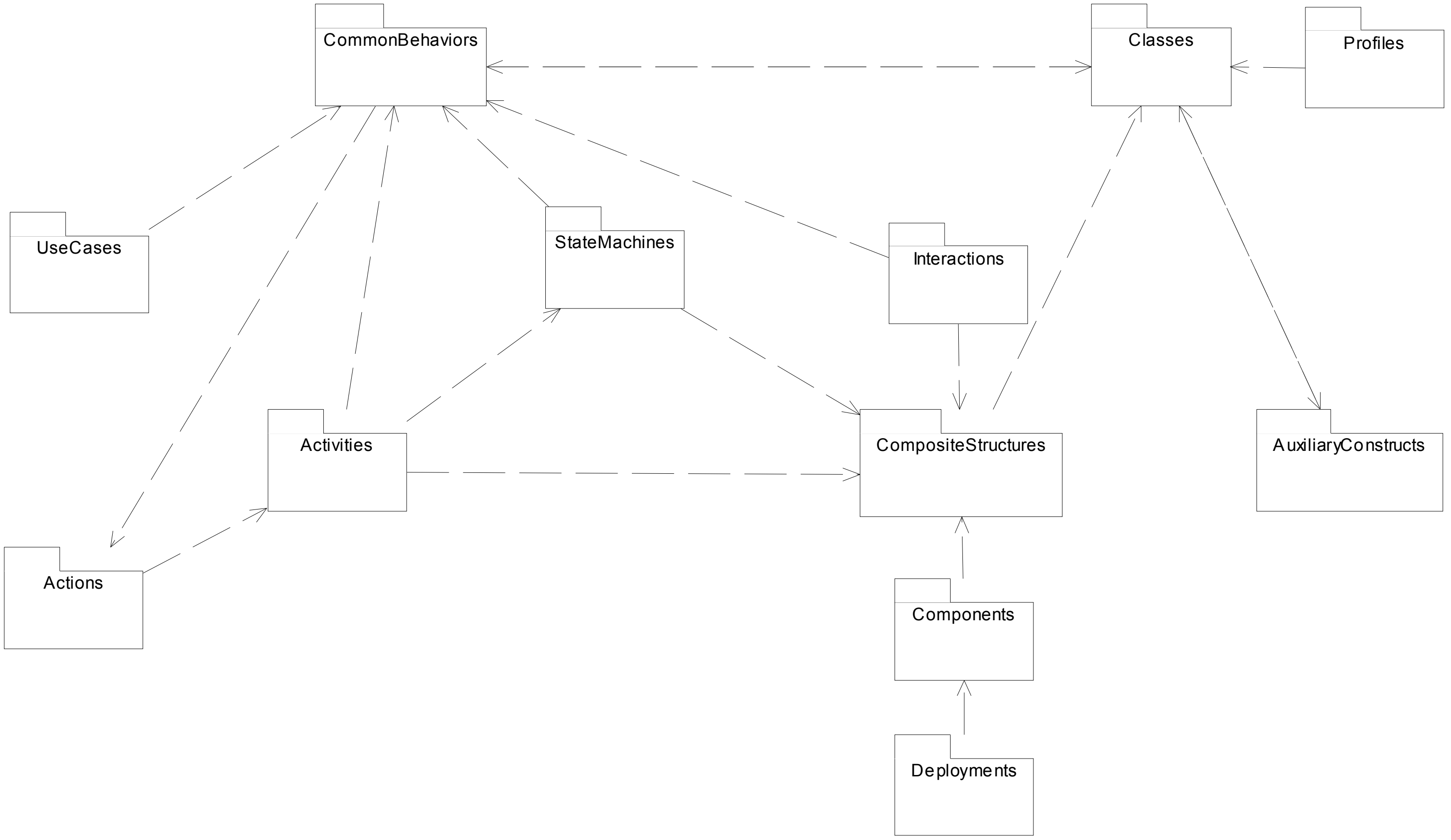


Language Architecture Top Level

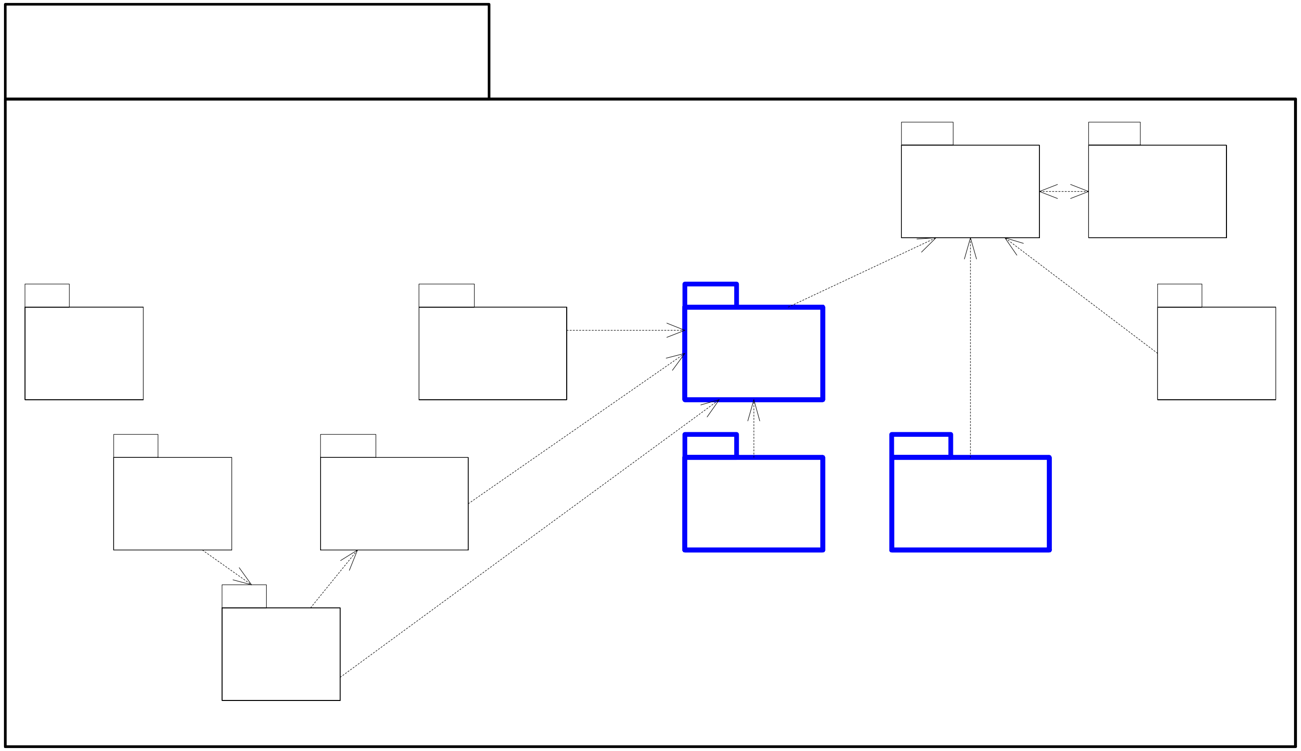


<<meta

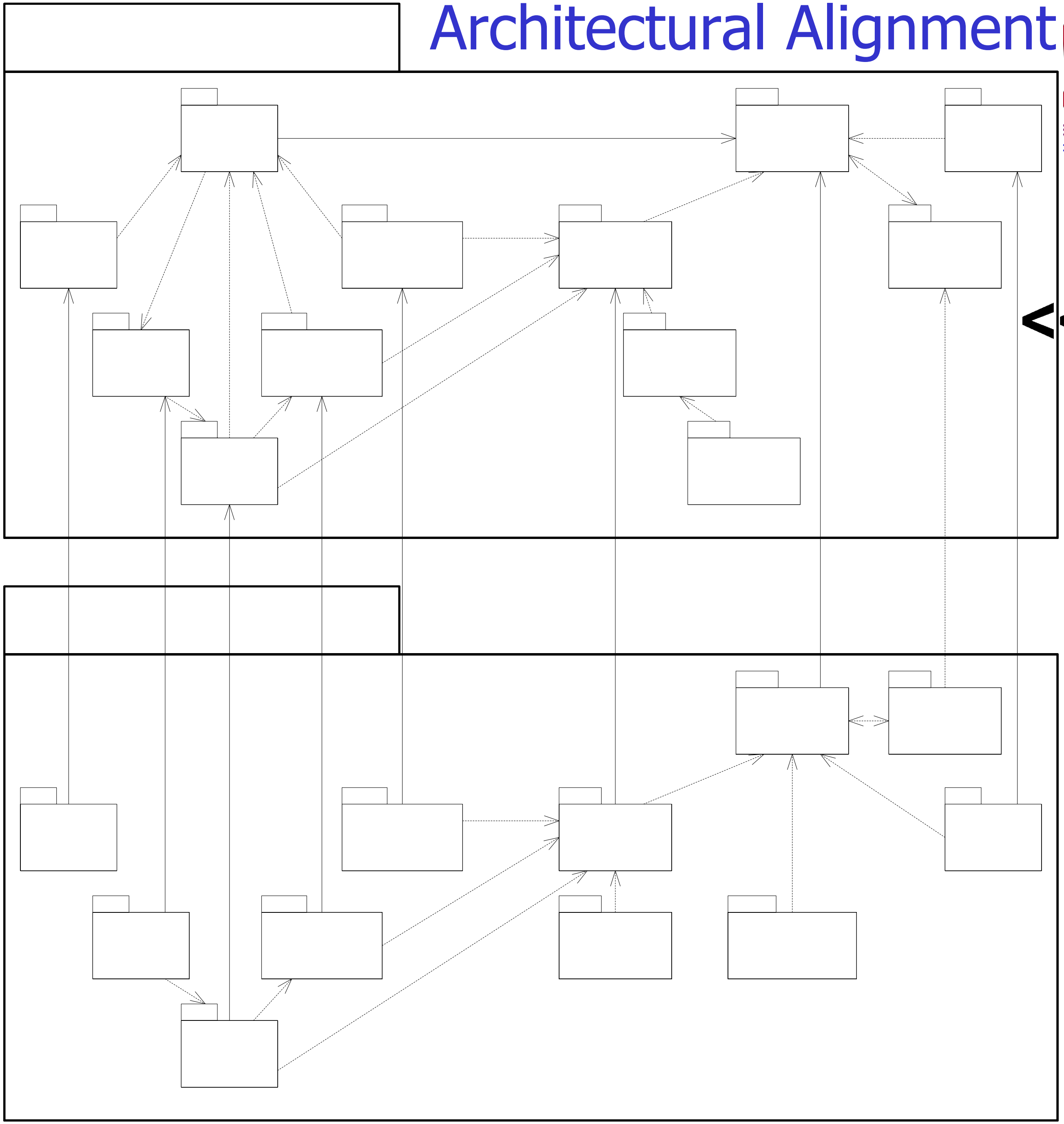
UML 2 Superstructure Architecture



SysML Language Architecture



Architectural Alignment



<<metamo

UML

Co

Beh

Extension Mechanisms

- Metamodeling
  - Subtyping the UML metamodel
  - Adding associations and attributes
- Stereotypes
  - Similar effect to subtyping the metamodel, but does not modify the repository schema
  - Cannot add new associations
- Model libraries
  - Like any other user model, except that they are standardized and available to be imported by any user

Profile = Stereotypes + Model Libraries + selective import of UML metamodel.

Major Extensions to UML 2

- Assembly Diagram
  - extends Composite Structure
  - enclosing class is an “assembly”
  - constraints on parts and ports
  - supports deep nested connectors
- Activity Diagram
  - accommodate needs of Extended Functional Flow Block Diagrams (EFFBDs)
  - extensions for continuous flow modeling
  - extensions to support disabling control and control operators

Other Extensions to UML 2

- Classes
  - extends properties to support specification of units and probability distributions on values
- Auxilliary
  - extends Information Items and Information Flows to include physical flows
  - adds primitive types for “real” and “complex”
  - specifies views and viewpoints
  - add model reference data

Major Extensions to UML 2 (cont.)

- Allocation
  - defines allocation relationship to allocate functions to components, etc
  - defines SysML::Deployment that integrates with assembly diagram
- New Diagram Types
  - Requirement Diagram
  - Parametric Diagram
- Allows for Other Diagram Usages
  - Context diagram (usage of class diagram)

Other Extensions Under Consideration

SYSTEMS MODELING LANGUAGE

- New diagram types and usages under consideration for future releases
  - Collaboration
  - Verification
  - Decision Tree
  - Causal Analysis

Underlying Semantic Model

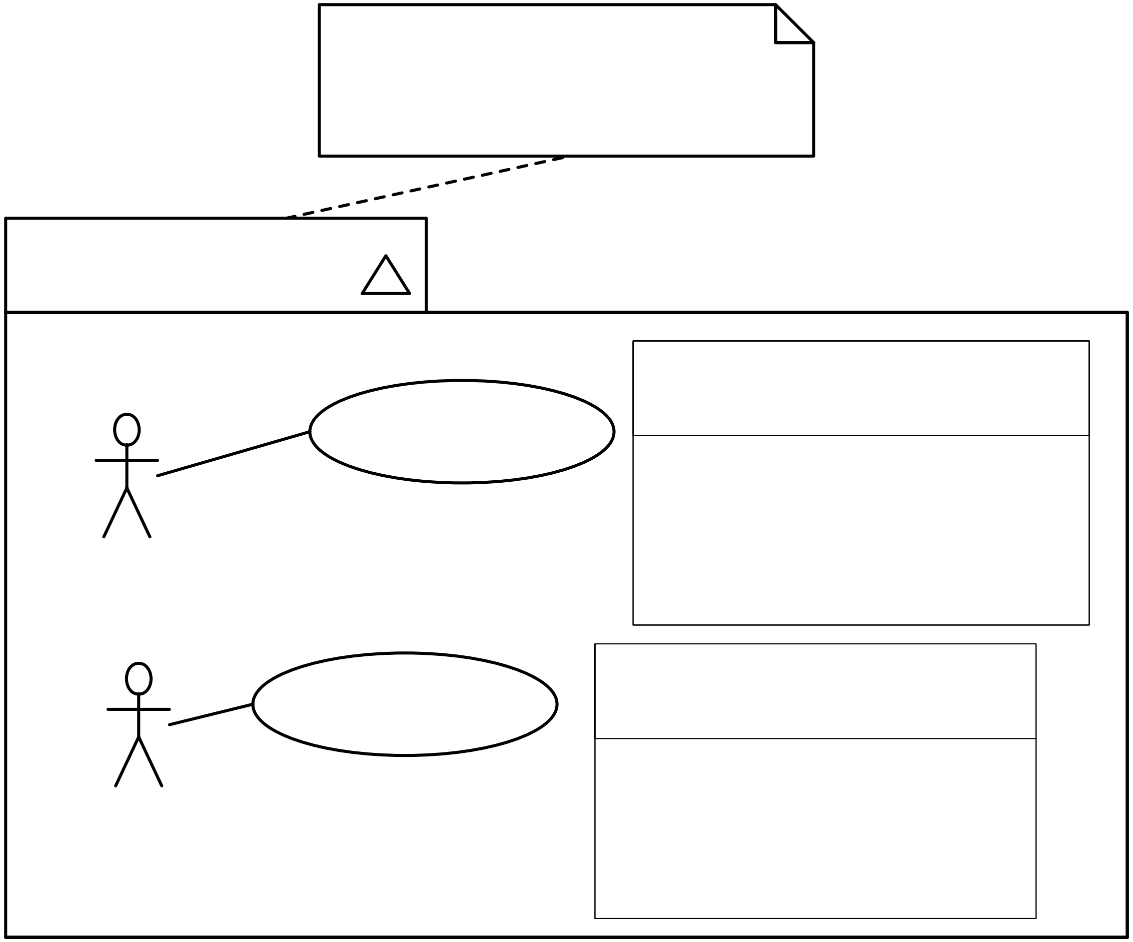
- UML 2 provides the underlying semantics that SysML builds upon
  - Semantic consistency defined by UML 2
- Methodology can enforce additional constraints to support further integration
  - SysML is intended to support multiple methodologies
- Tool vendors can implement constraints to enforce the methodology

Diagrams

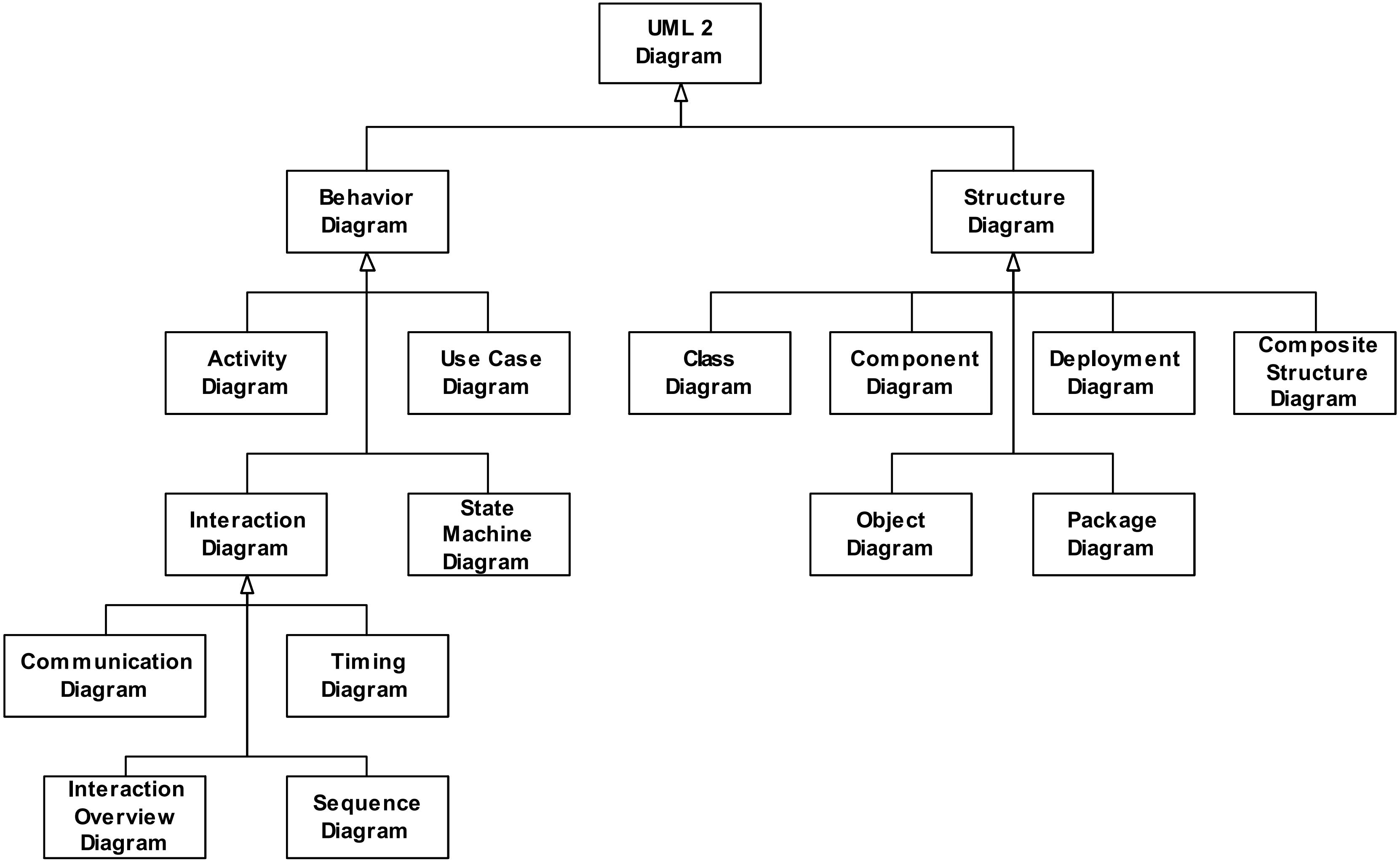
Models, Views, and Diagrams

- A model:
  - can be a metamodel and/or user model
    - a user model provides a representation (specification or characterization) of the physical system and its environment
  - can be decomposed into submodels
  - can include the semantics (abstract syntax) and/or notation (concrete syntax)
  - can be graphically represented by one or more diagrams
- A viewpoint is the perspective of a set of stakeholders that reflects the stakeholder concerns
- A view is a stereotype of a model that is intended to represent the model from a particular viewpoint

Viewpoint and View



UML 2 Diagram Taxonomy



SysML Diagram Taxonomy

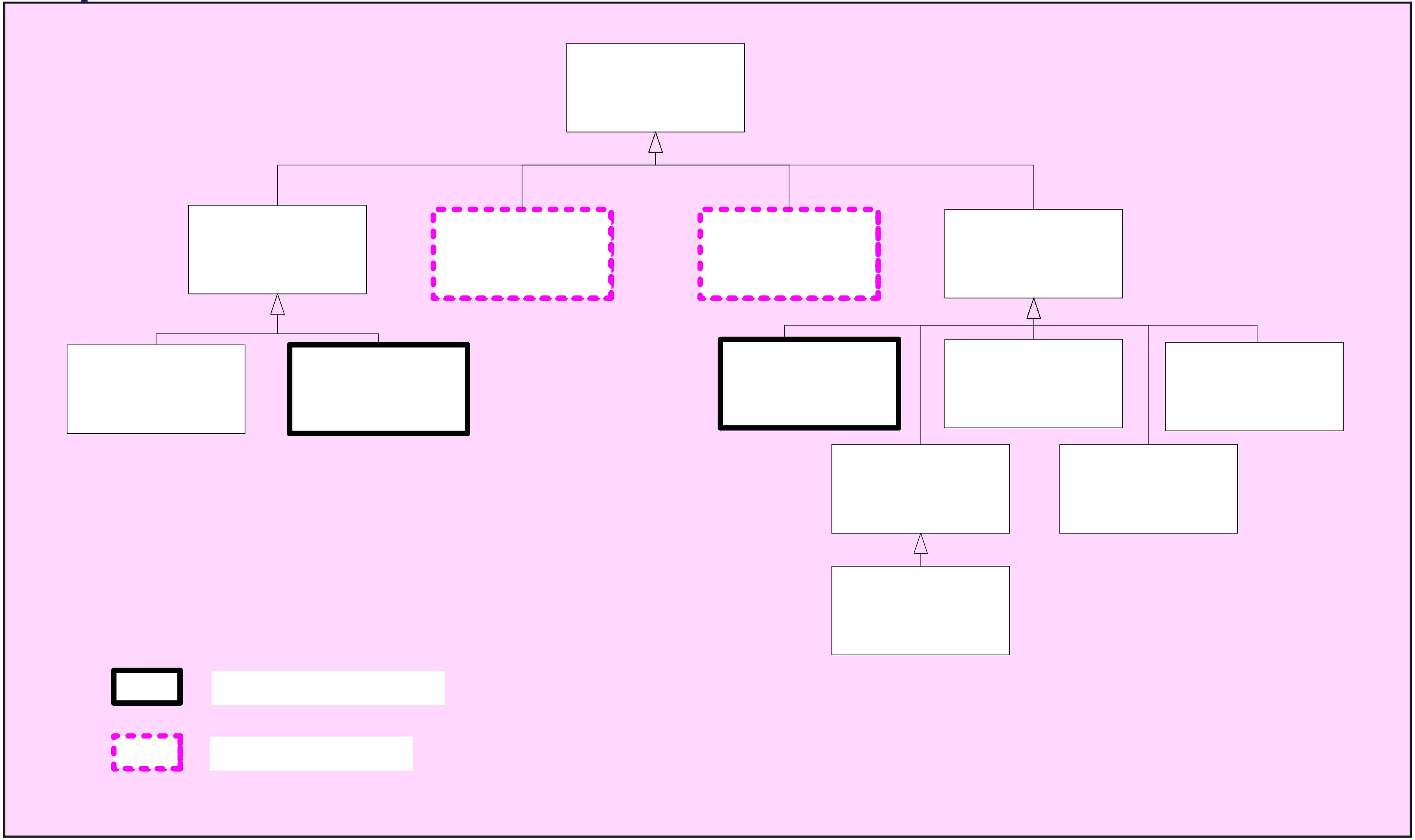


Diagram Frames

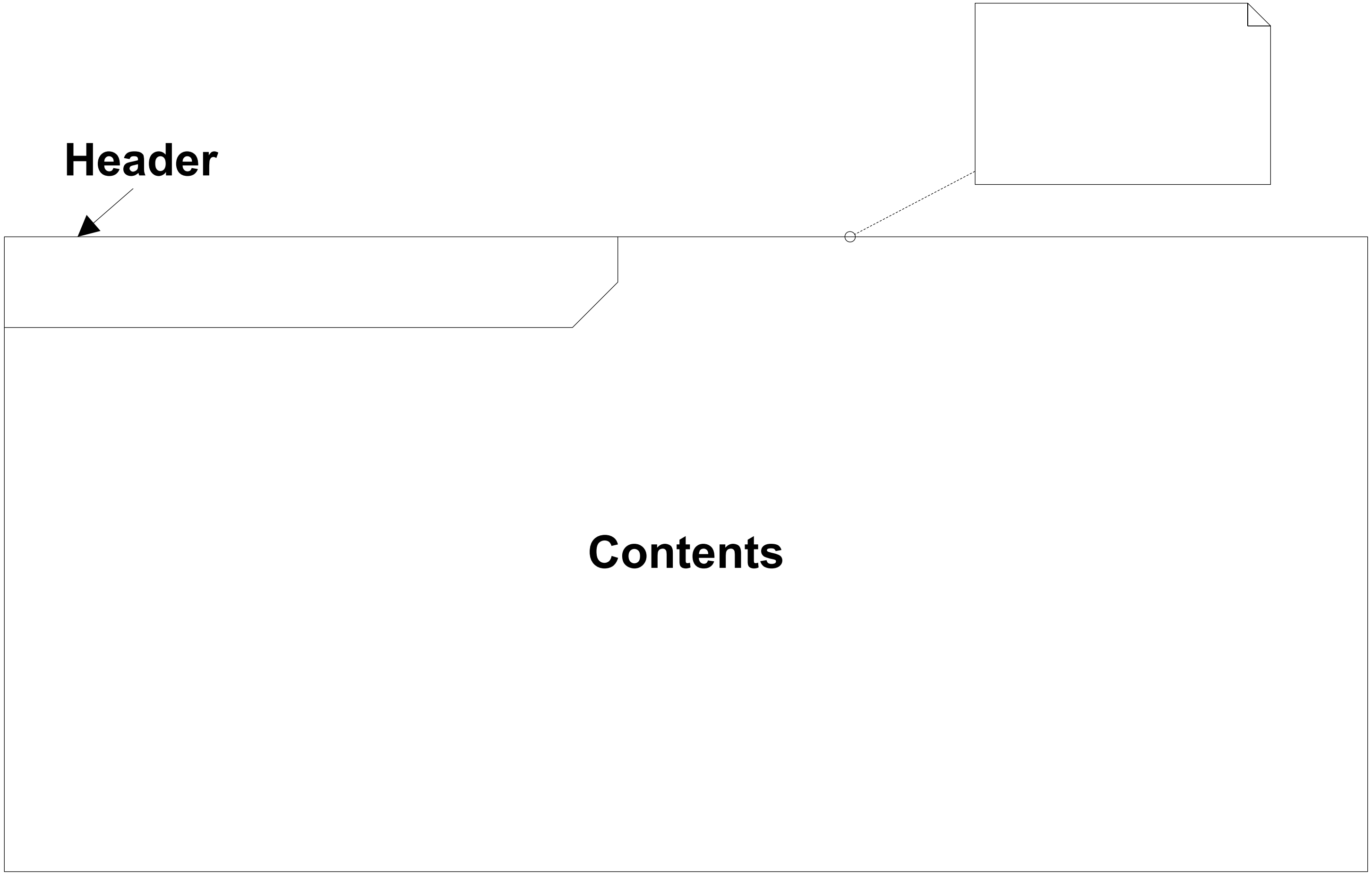
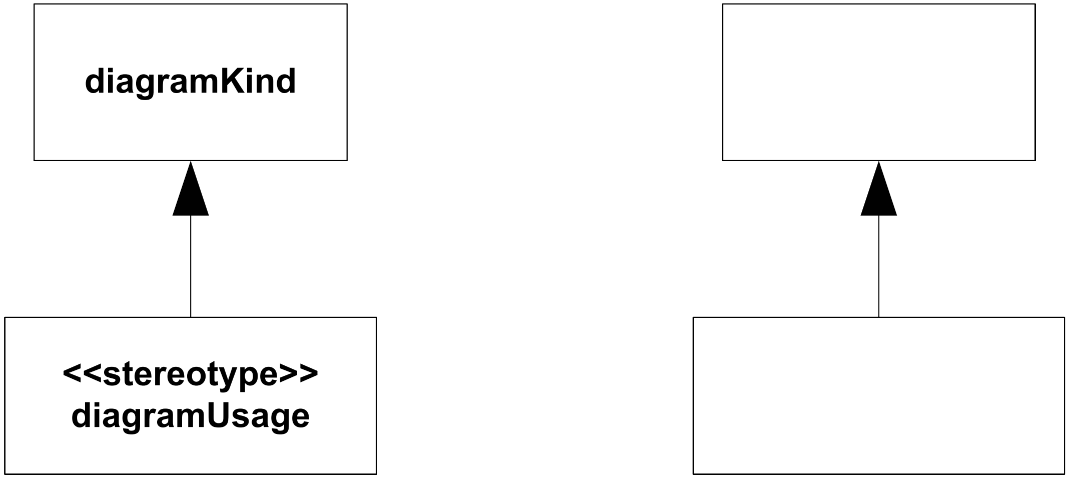


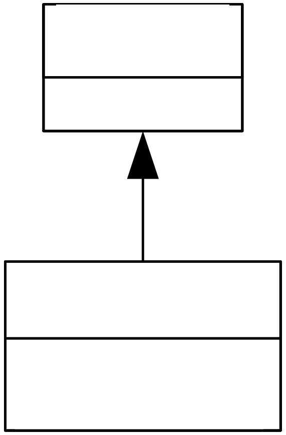
Diagram Usage

- Diagram usages can be added to the diagram taxonomy using the stereotype extension syntax
  - designated in the frame with guillemots



Model Element Reference Data

- Model Elements can include reference data
  - version
  - description
  - reference
  - user defined fields
- Reference data is a stereotype of comment
- Each field may include name, type, and value
- Represented by an extension of a UML comment
- Could be extended to include a version relationship between model element versions to support CM



Alternative Diagram Representations

- Alternative concrete syntax
  - graphical
  - tabular (optional)
  - tree (optional)
- Concrete syntax must conform to abstract syntax and constraints

Hybrid Diagrams

- Hybrid diagrams can include combinations of diagram types
  - May include a mix of structure, behavior, parametrics, and requirements
  - May be applied at different levels of nesting such as activities within states or at different levels of the system hierarchy

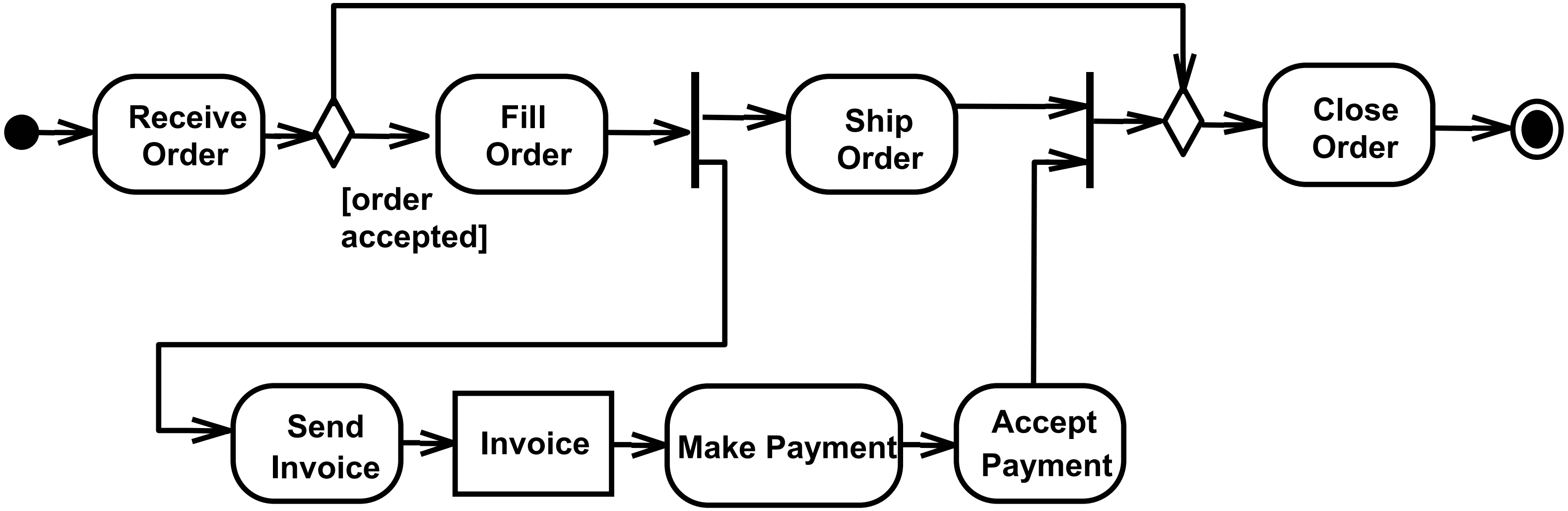
Activity Diagram
& Comparison With EFFBD

Activity Modeling

- Activity modeling emphasizes the sequence, inputs and outputs, and conditions for coordinating other behaviors (functions)
- Secondary constructs show which classifiers are responsible for those behaviors.
- Focuses on what tasks need to be done, in what order, with what inputs, rather than what performs each task

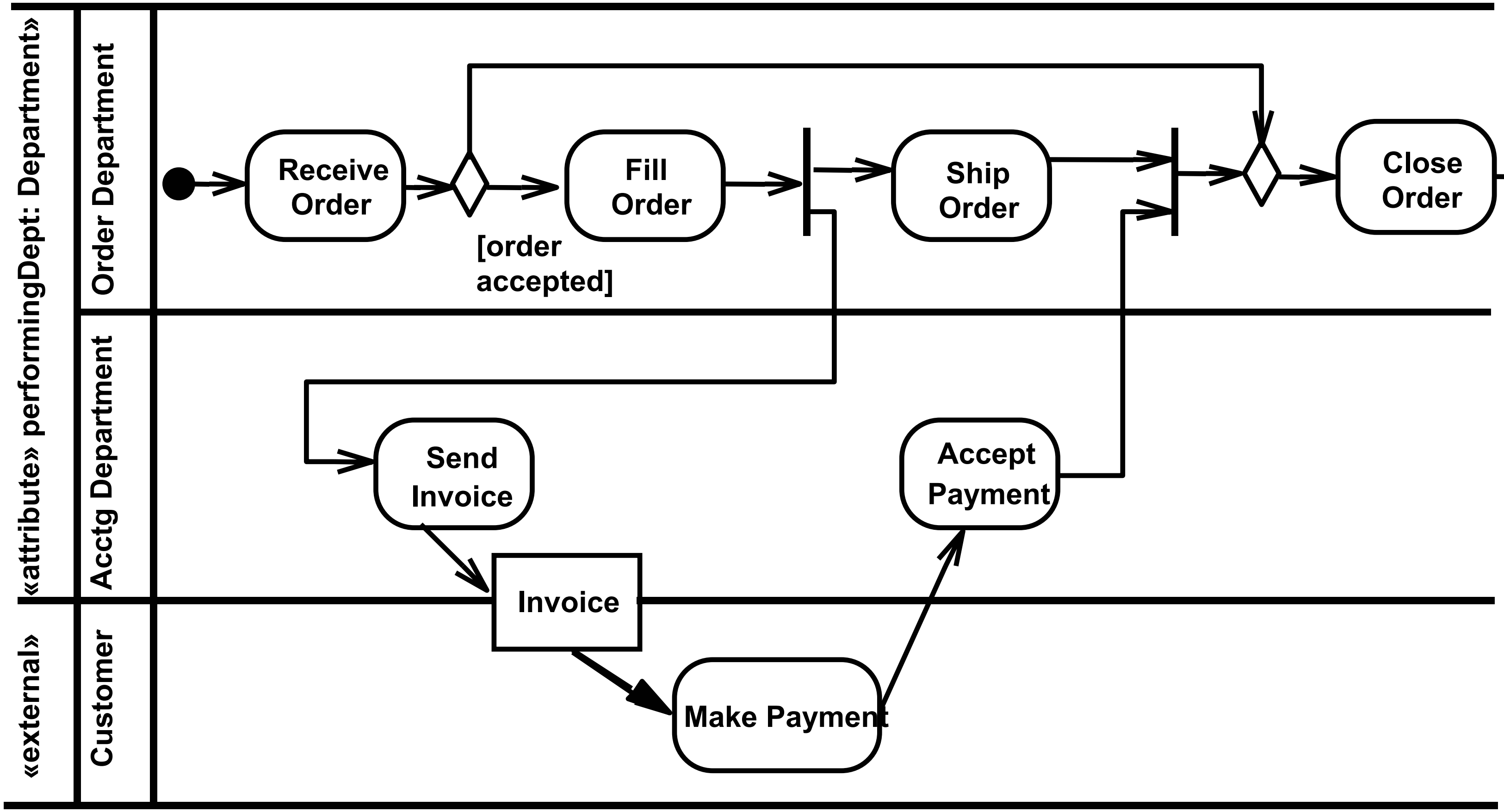
Activity Modeling

- Tasks and ordering …



Activity Modeling

- plus allocation to parts/assemblies via swim lanes (optional)



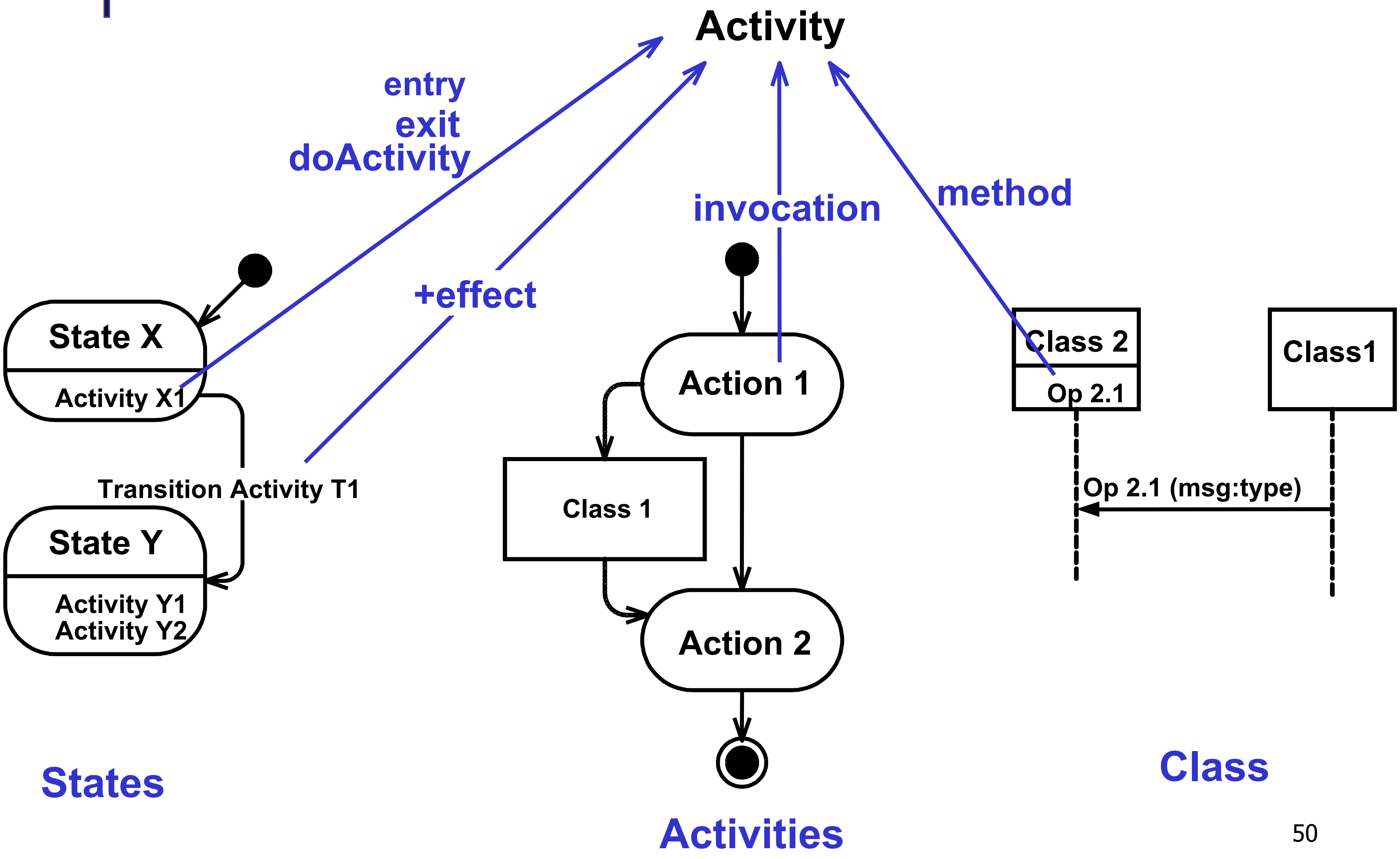
UML 2 Activities

- First-class behavior model:
  - usable with or without objects
  - parameterized
  - behavior properties
- Full parallelism
  - concurrent branches operate independently.
- Input/output
  - queuing, storage
  - notation
  - multi-entry/exit
- Full action model
  - model execution and simulation.

Activity Usage

- Activity is the a generic, reusable description of behavior
- Used in
  - other activities (decomposition through actions)
  - state machines
    - Transition activity
    - Entry/Exit activity on states
    - Do activity on states
  - Classes, Sequence Diagrams
    - Methods for operations
- Actions, states, and operations can reference the same activity
  - not envisioned to be standard practice

Activity Usage



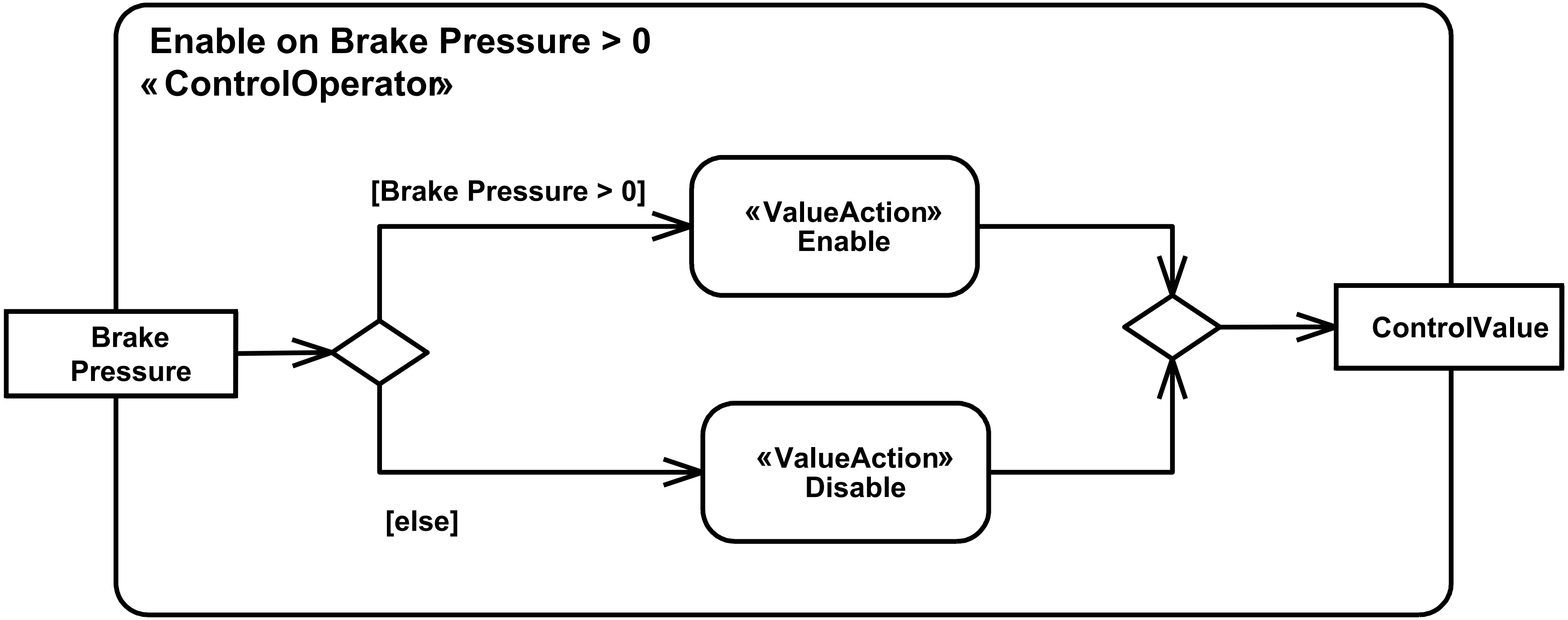
Relation Between Models

- The models emphasize different aspects of behavior
  - Activities: inputs, outputs, control
  - States machines: reaction to events
  - Operations: services of classes.
- Translation of activity and state models to sequence models
  - Actions specify when messages are sent.
  - Timing diagrams can be generated from model execution.

SysML Activity Extensions

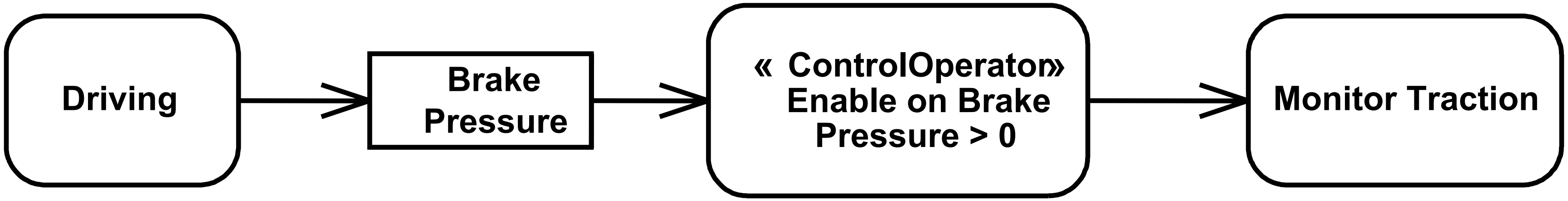
- Control as Data
  - Additional control values: disabling.
  - Control operators
- Continuous Systems
  - Flow rate
  - Updating stale data
  - Function patterns
- Functional Decomposition
- Probabilities
- Other extensions for EFFBD “profile”

Control as Data



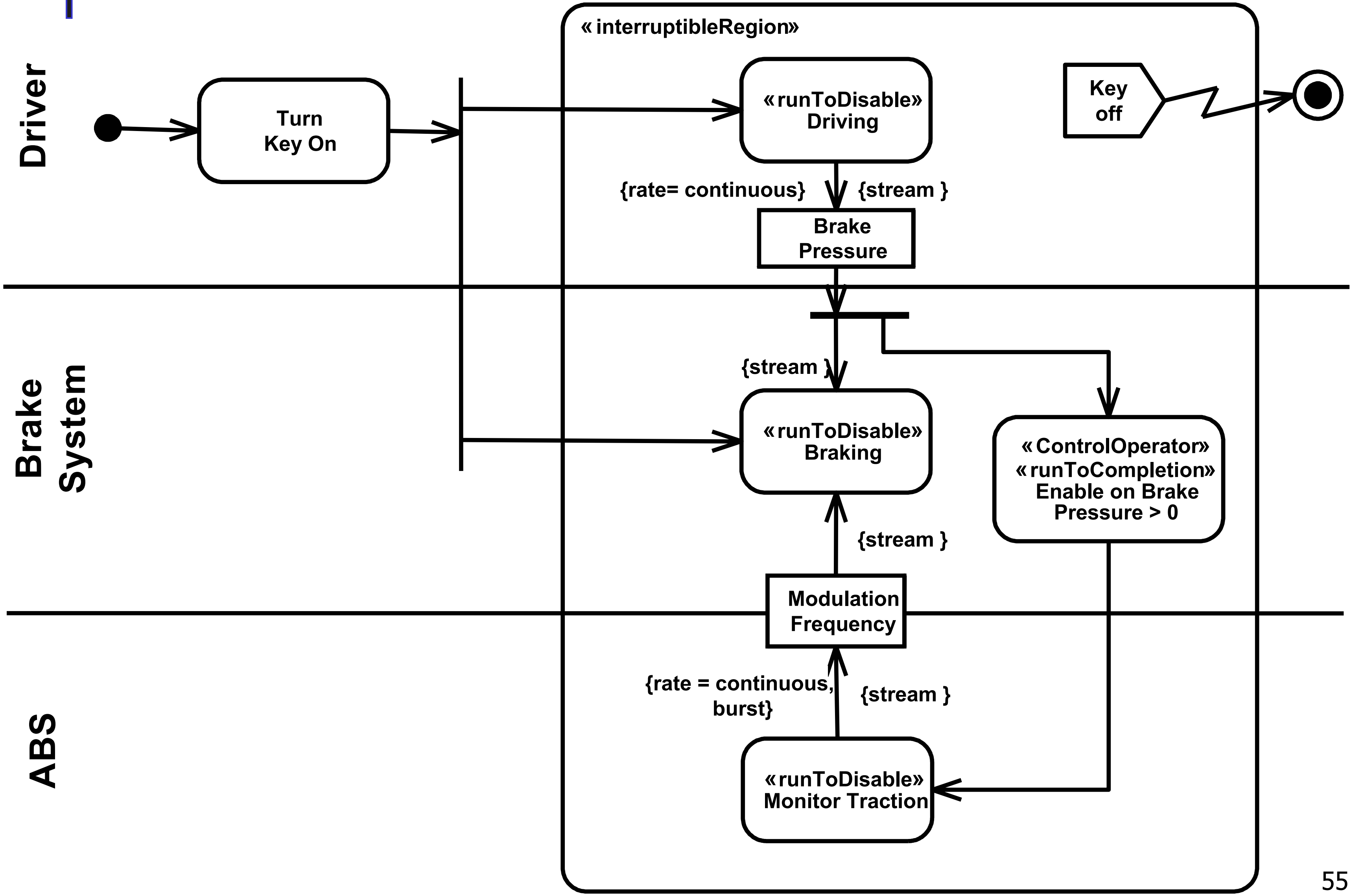
- Emits enabling or disabling control value based on input.

Control as Data

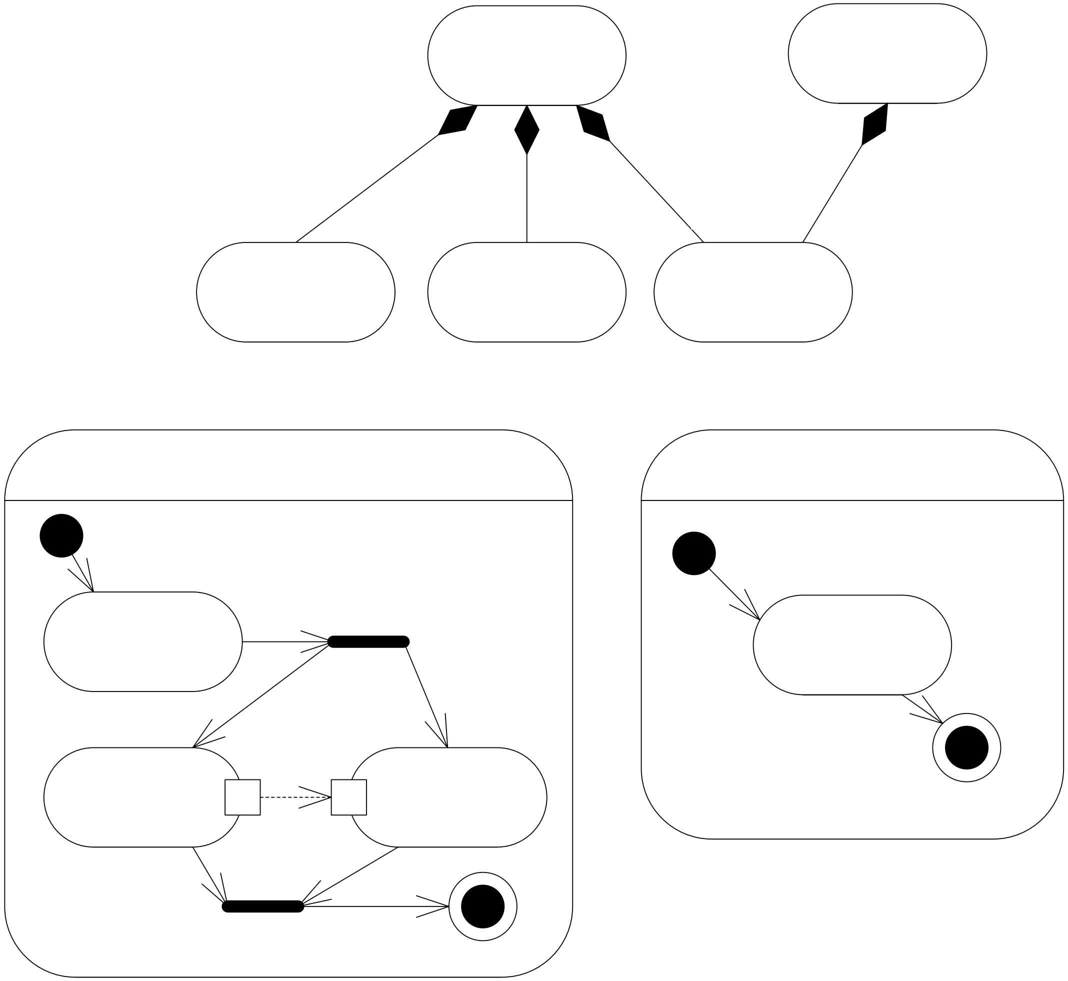


- Brake Pressure determines when traction is monitored in ABS sytem.

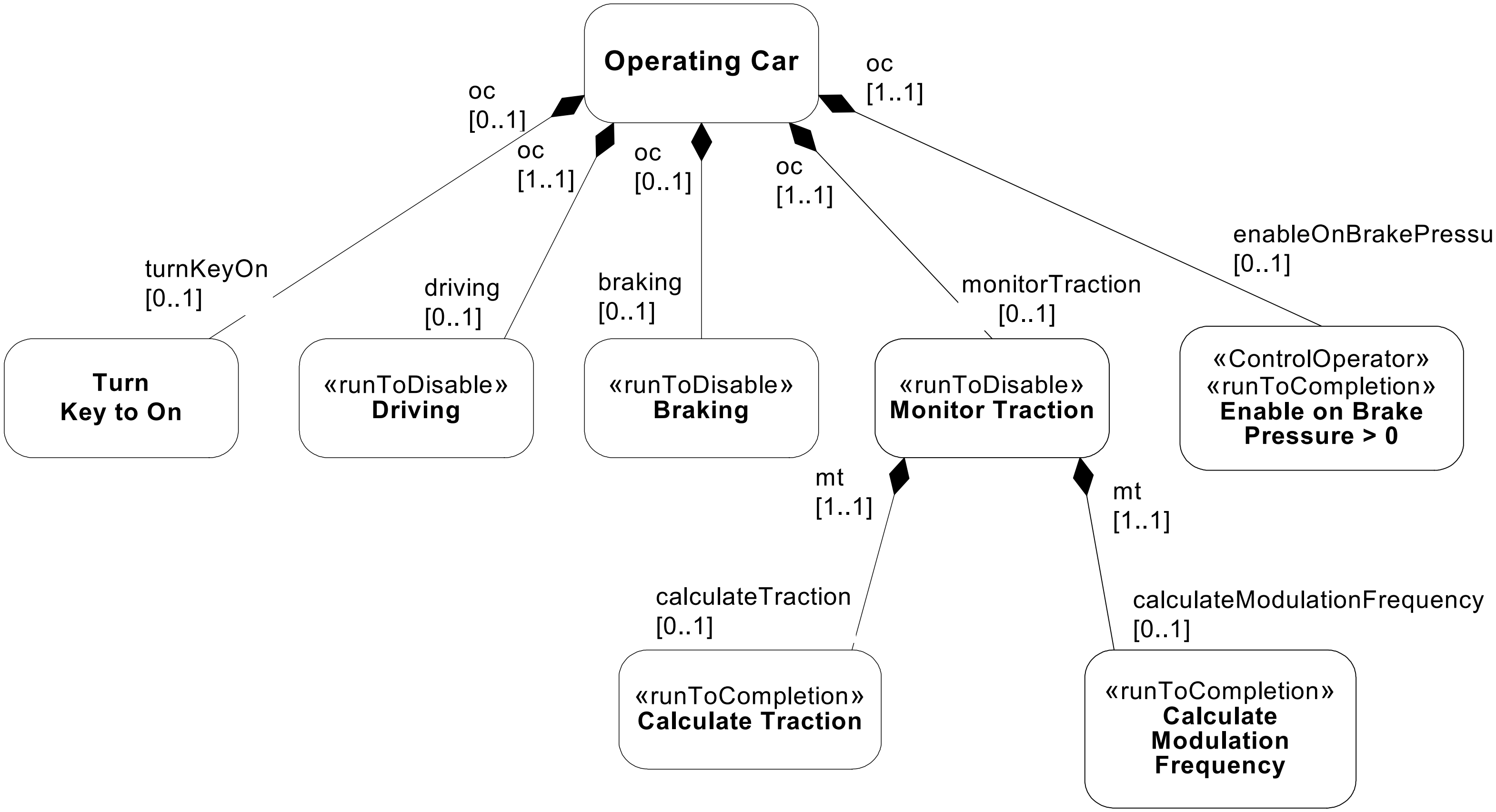
Continuous Systems



Functional Decomposition (cont.)



Functional Decomposition (cont.)



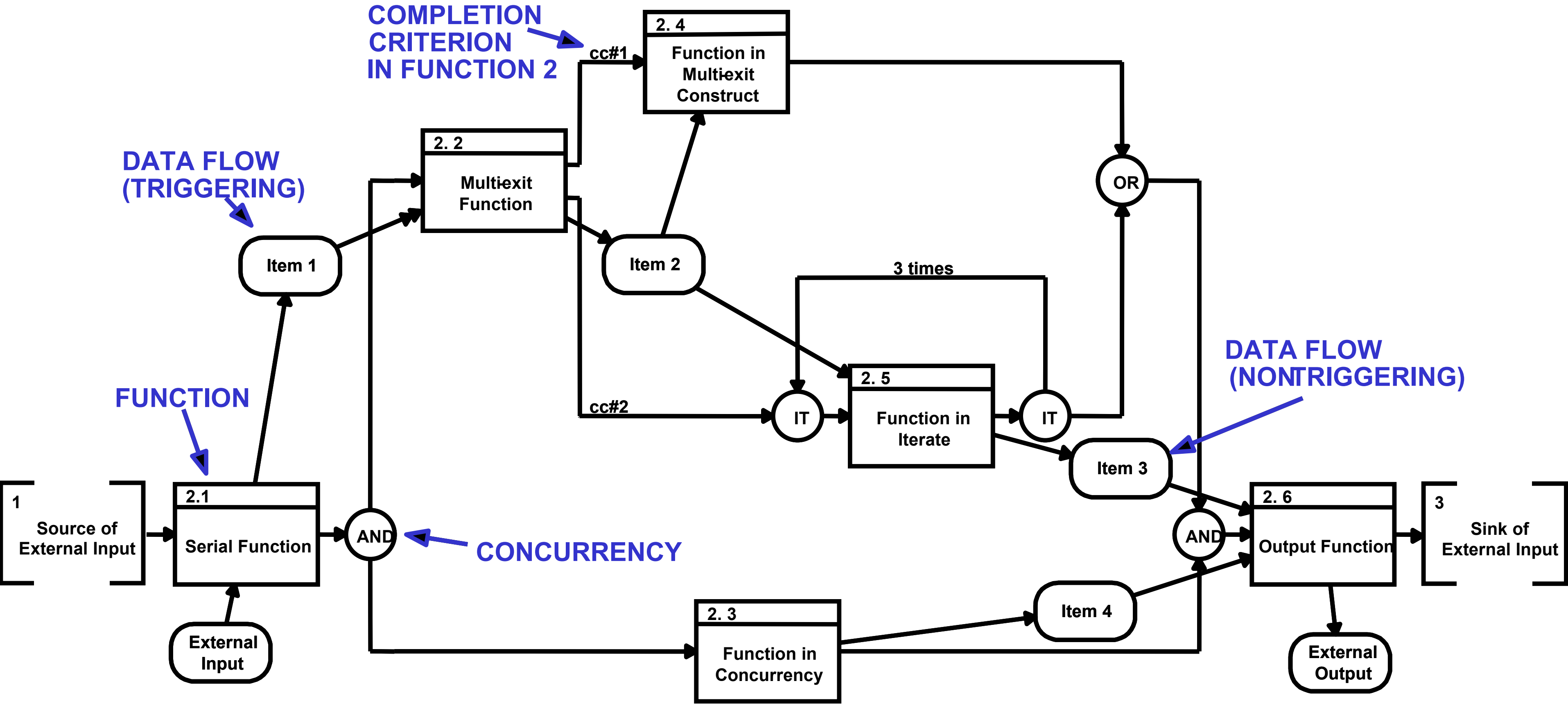
Probabilities

- Flows from decisions nodes.
  - Can refer to root of decision tree.
- Parameter sets (= EFFBD multi-exit functions)
- Properties and parameters
  - Over a single instance or execution over time
  - or over all instances of a class and or executions of a behavior.

Extensions for EFFBD

- Control parameters (for multi-exit functions that output control).
- Control pins (for control queuing)
- TBD: Replication

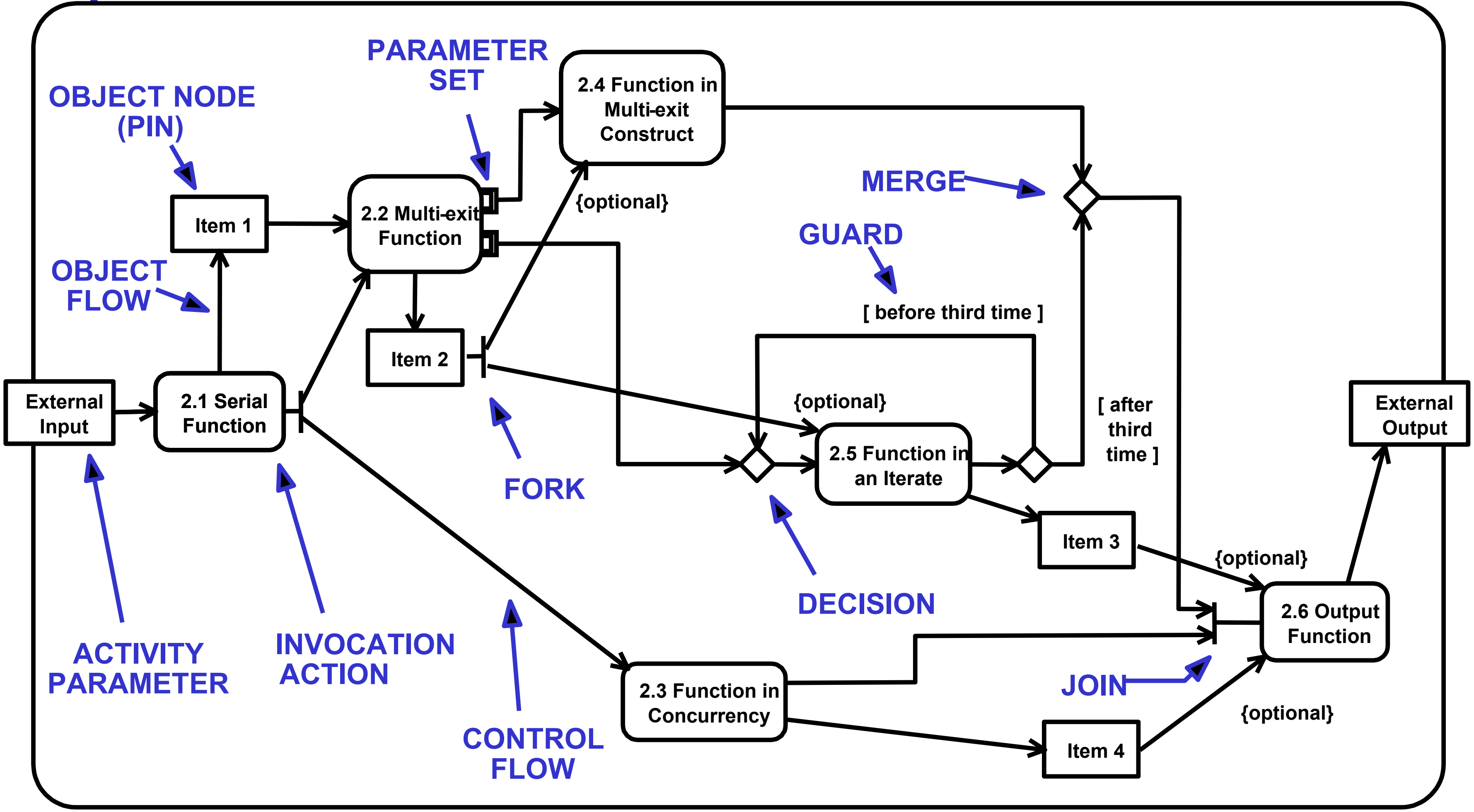
Extended Functional Flow Block



Adapted from Long, J., "Relationships between Common Graphical Representations in System Engineering", ViTech Corporation, www.vitechcorp.com

EFFBD ⇔ UML

SysML™ SYSTEMS MODELING LANGUAGE



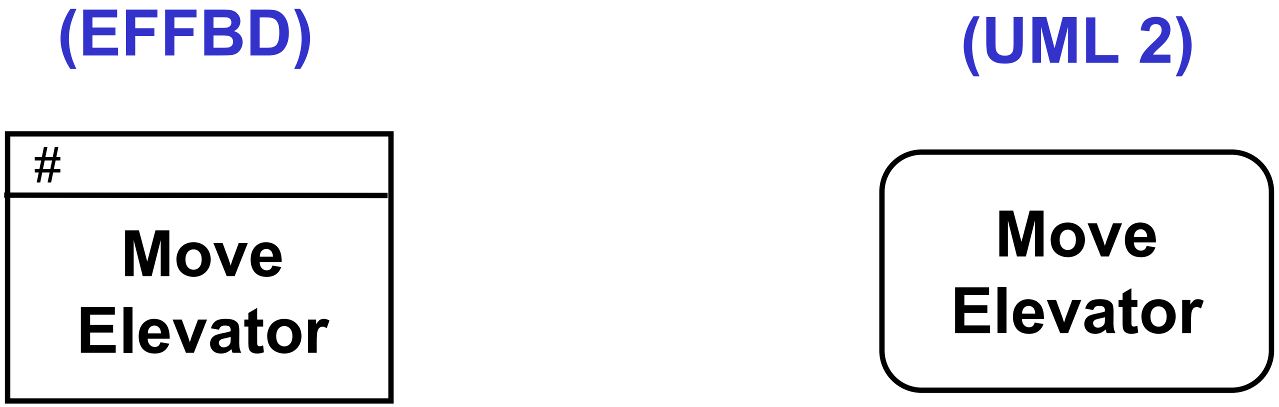
From Bock, C., "UML 2 Activity Model Support for Systems Engineering Functional Flow Diagrams," Journal of INCOSE Systems Engineering, vol. 6, no. 4, October 2003.

EFFBD ⬄ UML

- Triggering Item Input⬄ Required Input
- Nontriggering Input ⬄ Optional Input
- Select ⬄ Decision, Merge
- Branch Annotation ⬄ Guard
- Concurrency ⬄ Fork, Join
- Multi-exit Functions ⬄Activity with Output Parameter Sets
- Completion Criteria ⬄ Postconditions on Parameter Sets

Function ⇔ Behavior/Action

- EFFBD Function and UML 2 Action/Behaviors are steps in a process flow.



(EFFBD)

#

Move Elevator

(UML 2)

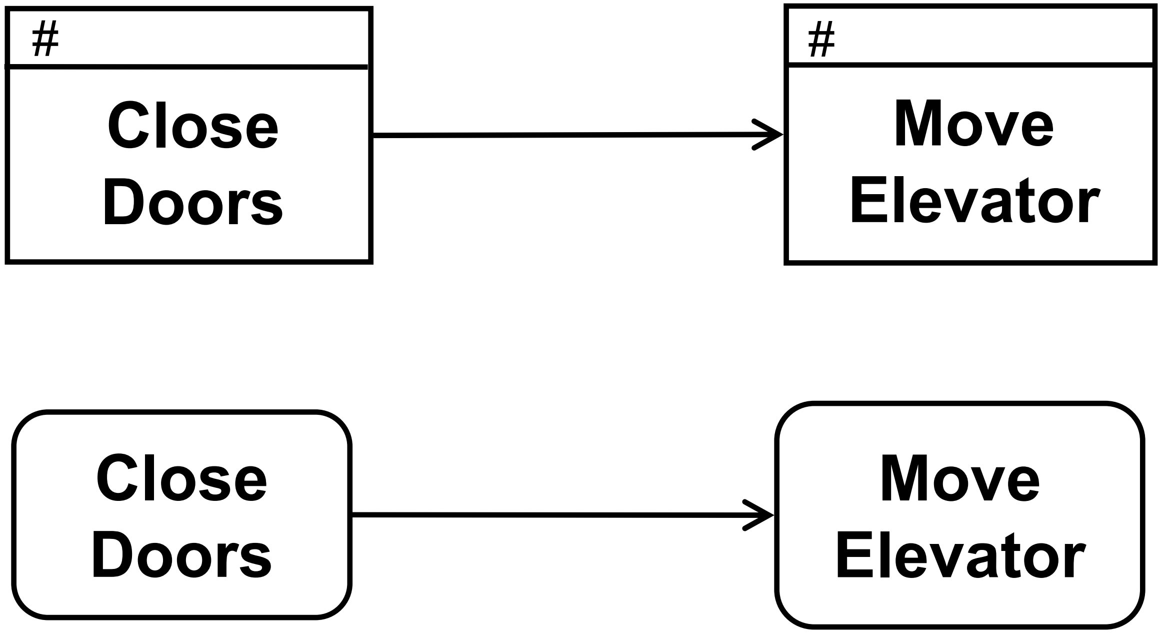
Move Elevator

- **Notation is different, but repository would be the same (except for adding #).**

Control Flow

- EFFBD and UML 2 Control Flow give time sequence to steps in a process flow.

(EFFBD)



#

Close Doors

#

Move Elevator

(UML 2)

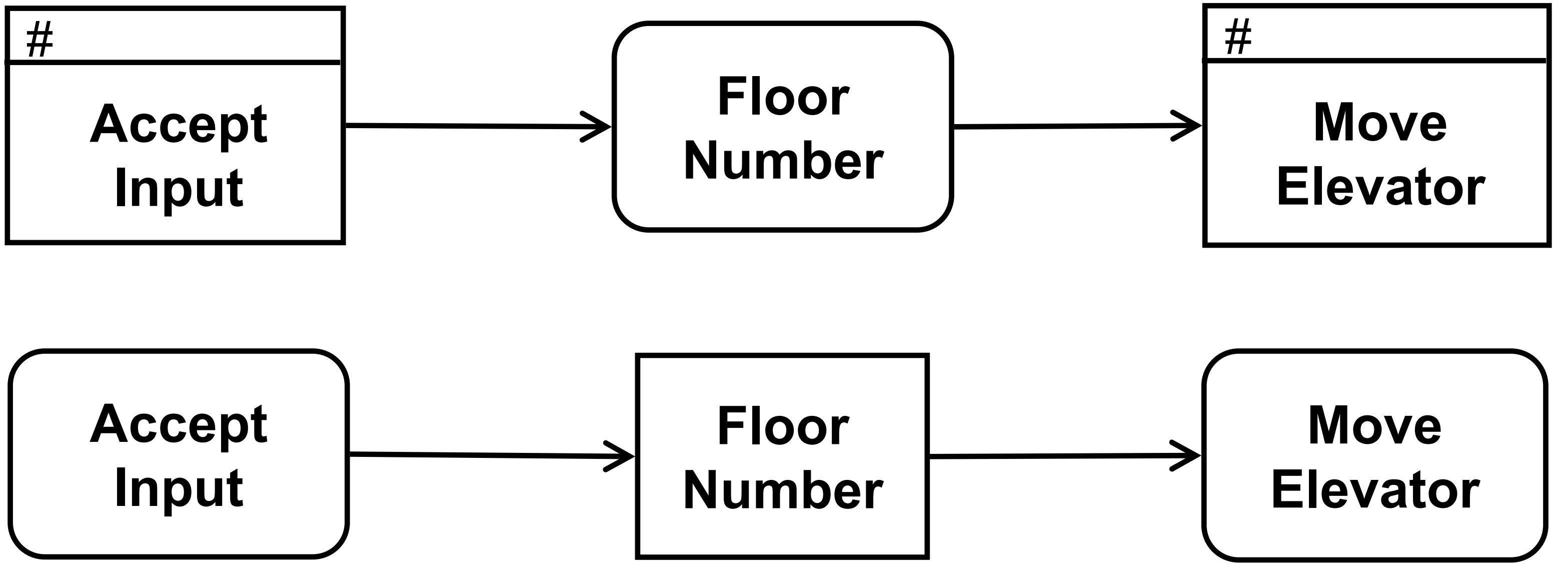
Close Doors

Move Elevator

Data/Object (Item) Flow

- EFFBD and UML 2 Data Flow specify how Function/Behavior outputs are provided to inputs.

(EFFBD)

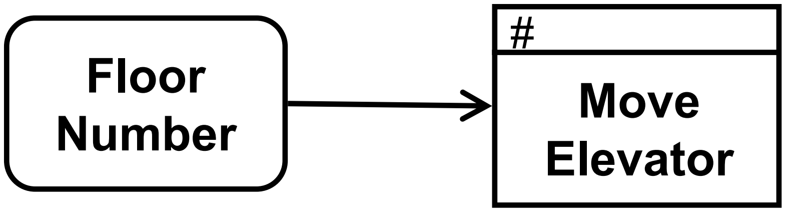


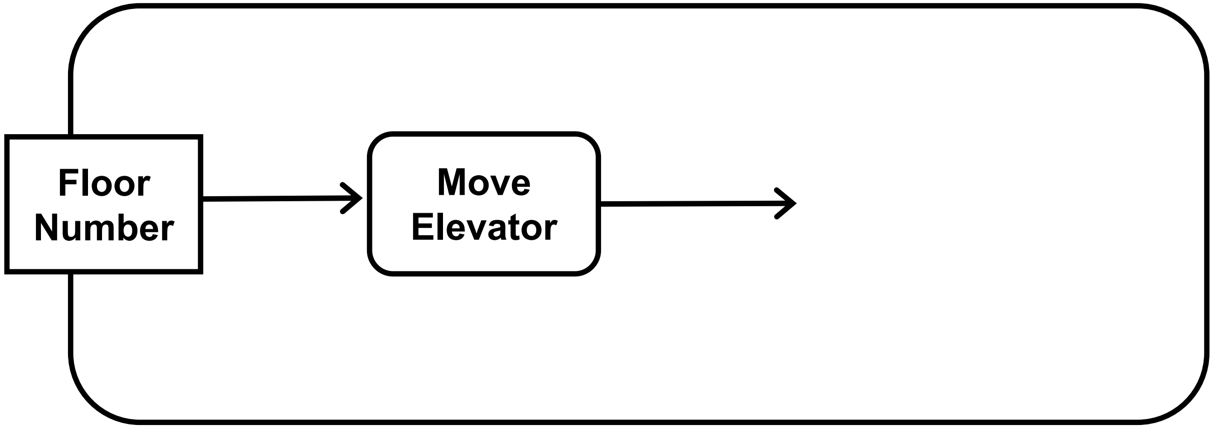
(UML 2)

External I/O ⇔ Parameter

- EFFBD External Input/Output and UML 2 Parameter support I/O at the beginning/end of the entire diagram.

(EFFBD)



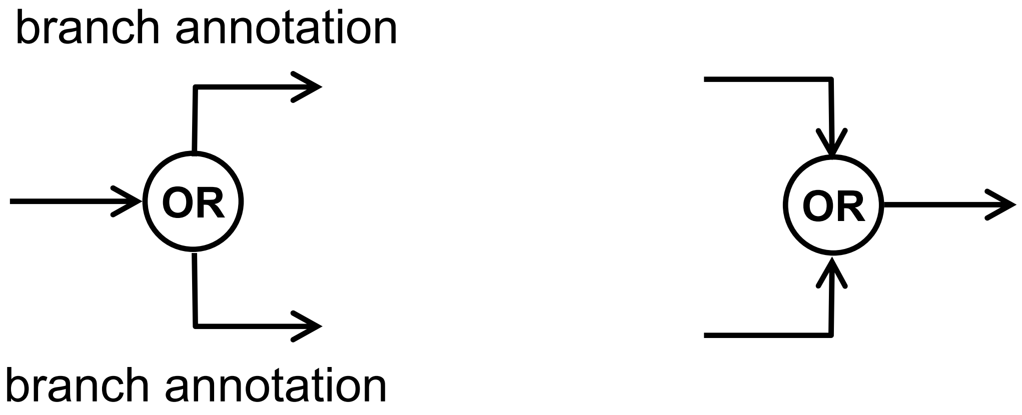


(UML 2)

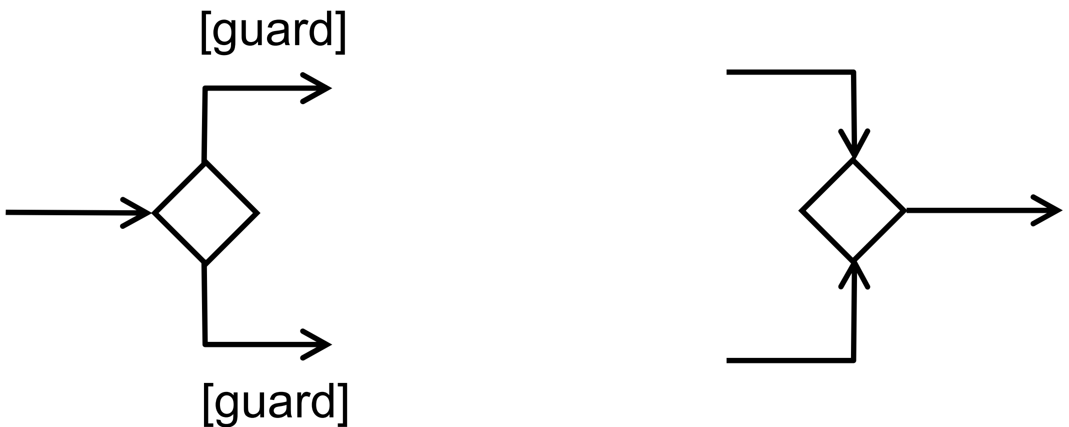
Select ⇔ Decision

- EFFBD Select and UML 2 Decision specify mutually exclusive paths in a flow.

(EFFBD)

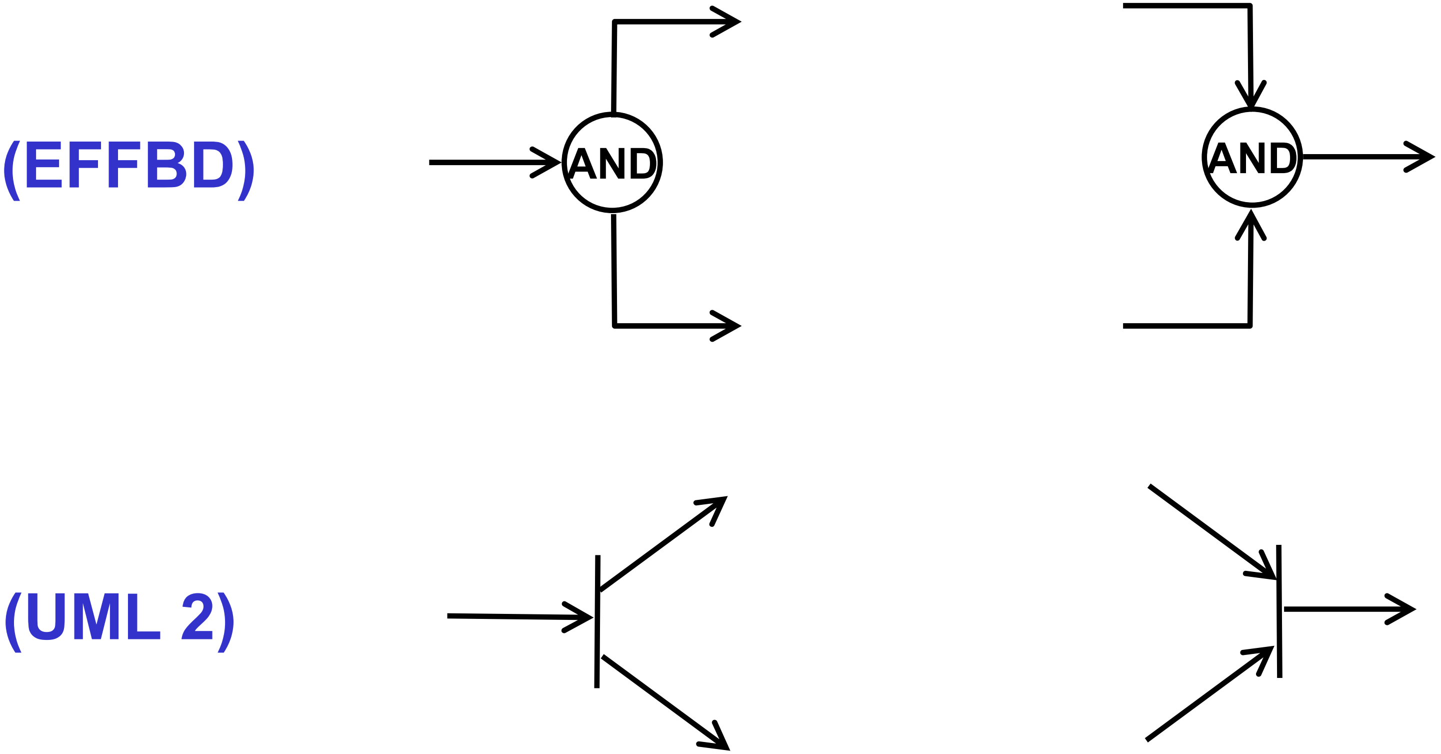


(UML 2)



Concurrency ⇔ Fork/Join

- EFFBD Concurrency and UML 2 Fork/Join specify parallel paths



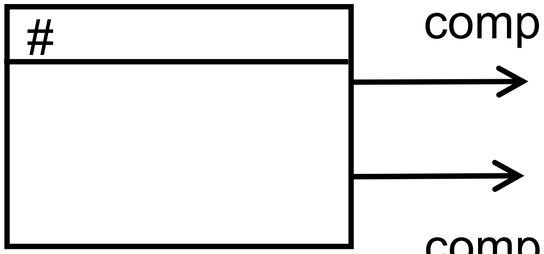
(EFFBD)

(UML 2)

Multi-exit ⇔ Parameter Sets

- EFFBD multi-exit functions and UML 2 Parameter Sets specify mutually exclusive outputs.

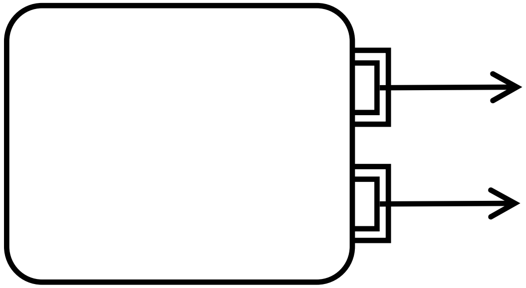
(EFFBD)



completion condition

completion condition

(UML 2)

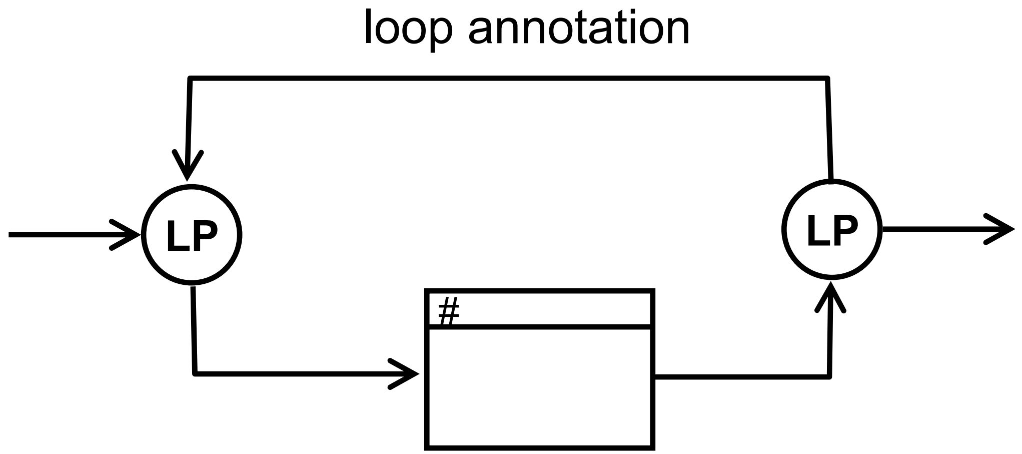


TBD: Postcondition
on parameter set

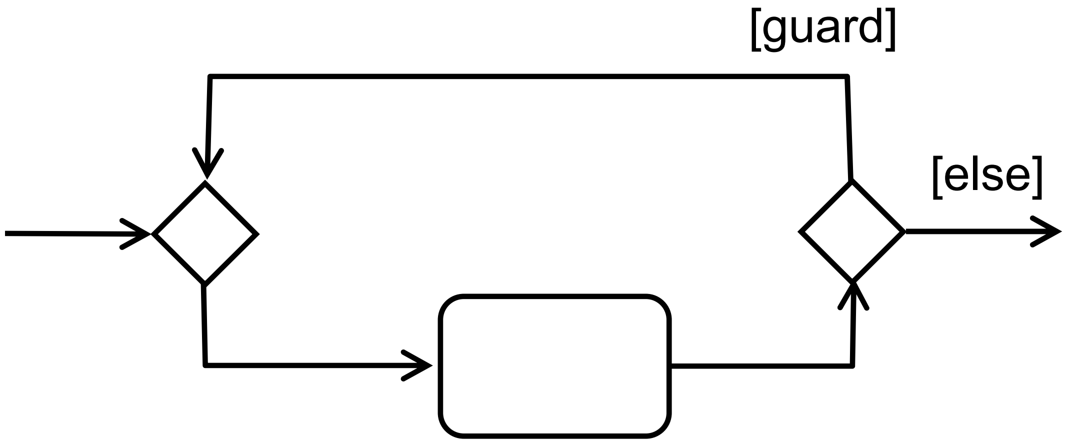
Cycles

- EFFBD and UML 2 flows can have cycles in the flow graph.

(EFFBD)



(UML 2)



Action Execution Rules

- **Before execution**
  - an action waits for all required, non-streaming data/item inputs, in whatever quantity is required, and waits for all control inputs required, then begins.
  - Note: does not include rules requiring availability of resources
- **During execution**
  - data/objects/items arriving at non-streaming inputs while the action is executing are queued.
  - actions support concurrent execution of queued inputs (reentrancy). This should be compared to replication.
  - data/objects/items arriving at streaming inputs are input to the action execution.
  - control arriving is queued if control pins are used, otherwise, control values arriving at an action already executing are discarded. (except runToDisable actions respond to disable control value)
  - an action can provide streaming outputs.
- **After execution**
  - an action provides all required, non-streaming data/item outputs, in whatever quantity is required, and all control outputs required.

EFFBD Profile

- All actions require at least one control edge coming into them and going out.
- All forks have a corresponding join.
- The EFFBD OR notation is inclusive (though implementations are exclusive). It translates to a UML fork.
- Iteration indicators on decision nodes.
- All control flows are queuable.
- All parameters are nonstreaming.
- Double arrowheads for required inputs.
- UML translations provided for EFFBD constructs such as kill branches.

Assembly Diagram

SysML Assemblies

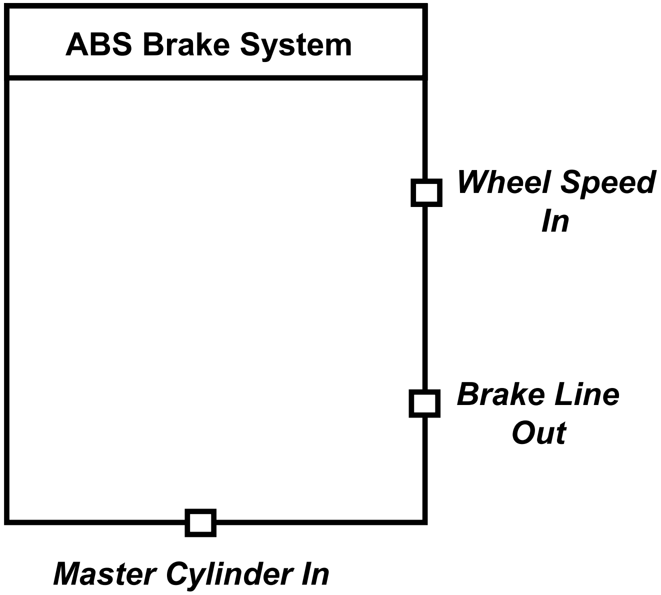
- UML 2 “component” is intended for modeling of software components
- SysML replaces the UML 2 “component” with a domain neutral concept of “assembly”, with relatively minor changes to UML 2 Composite Structures
- Assemblies do not allow UML 2 Interfaces (operation based interfaces and lollipop notation) targeted for the software domain
  - This concept of interfaces is considered to restrictive for general systems modeling

Structural Modeling Foundation

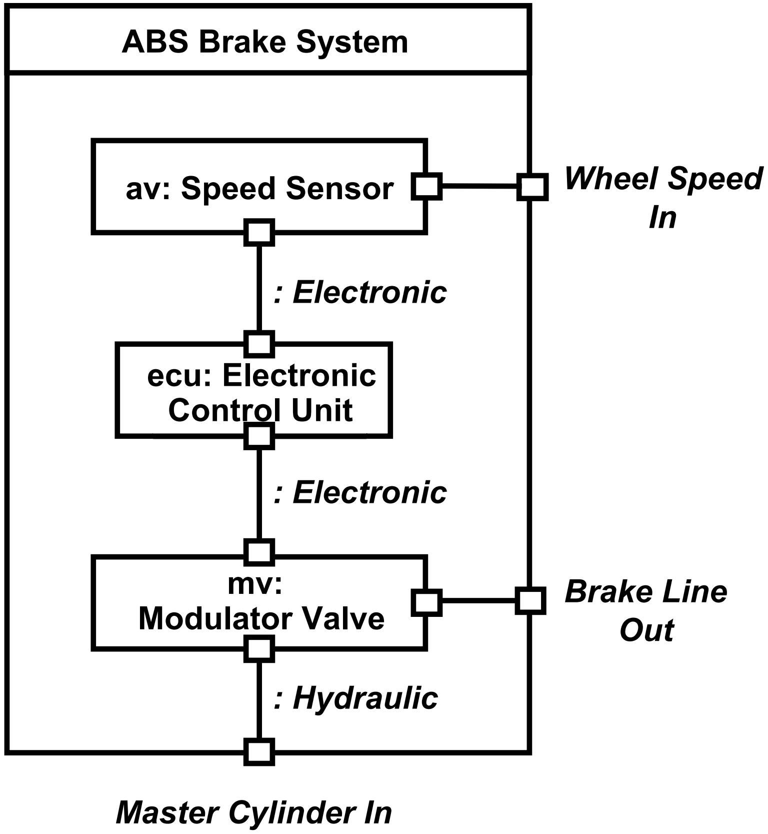
- Assemblies are structured classes, extended with an ability to hold ports, parts, and internal connectors
- "Assembly" captures a module at any level in the system hierarchy.
  - Can represent external systems, system of interest, logical, physical, hardware, software, etc.
  - Assemblies provide both black box view (without internal structure) and white box view (showing internal parts and connectors)

White vs. Black Box Views

Black Box



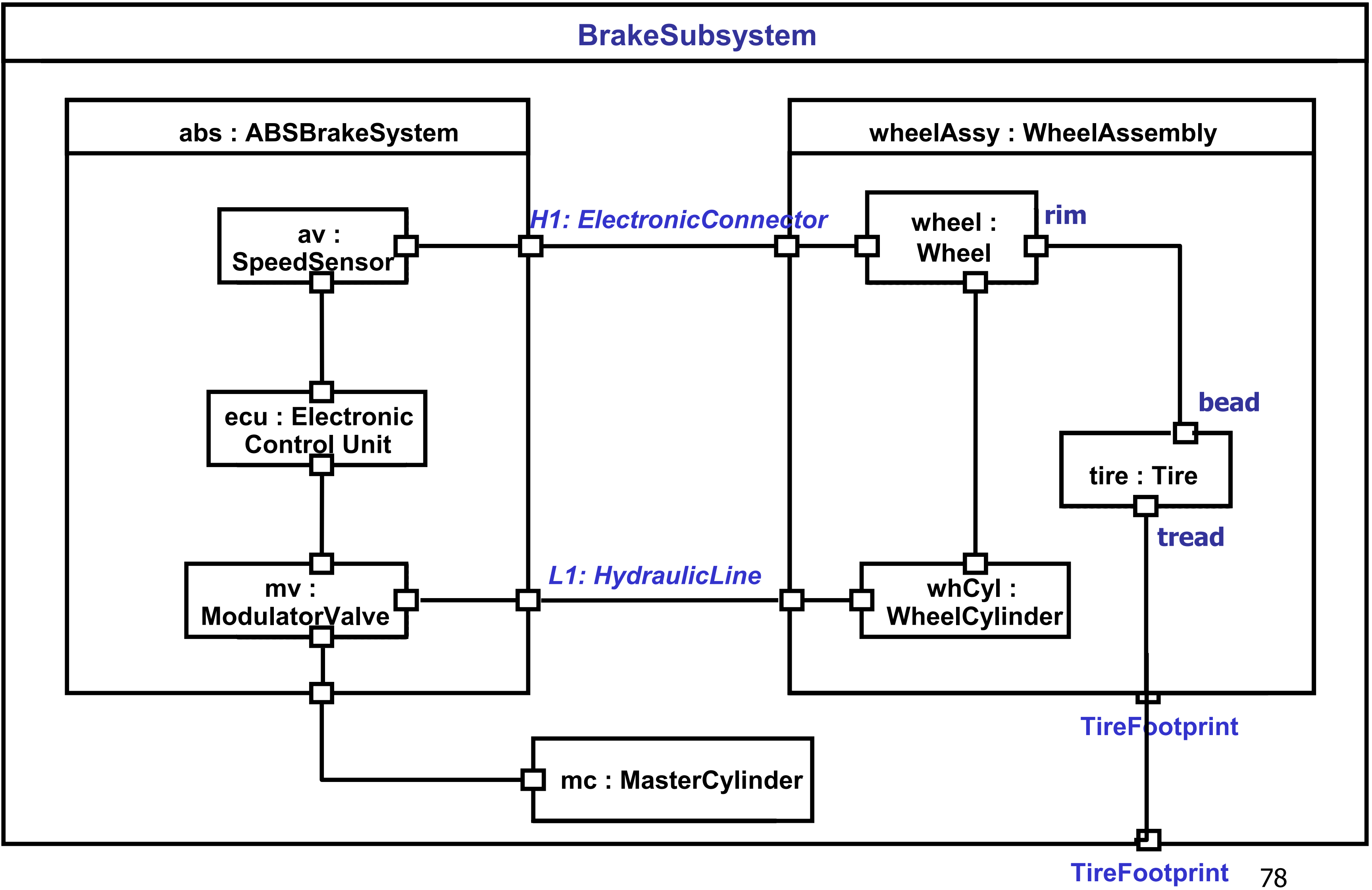
White Box



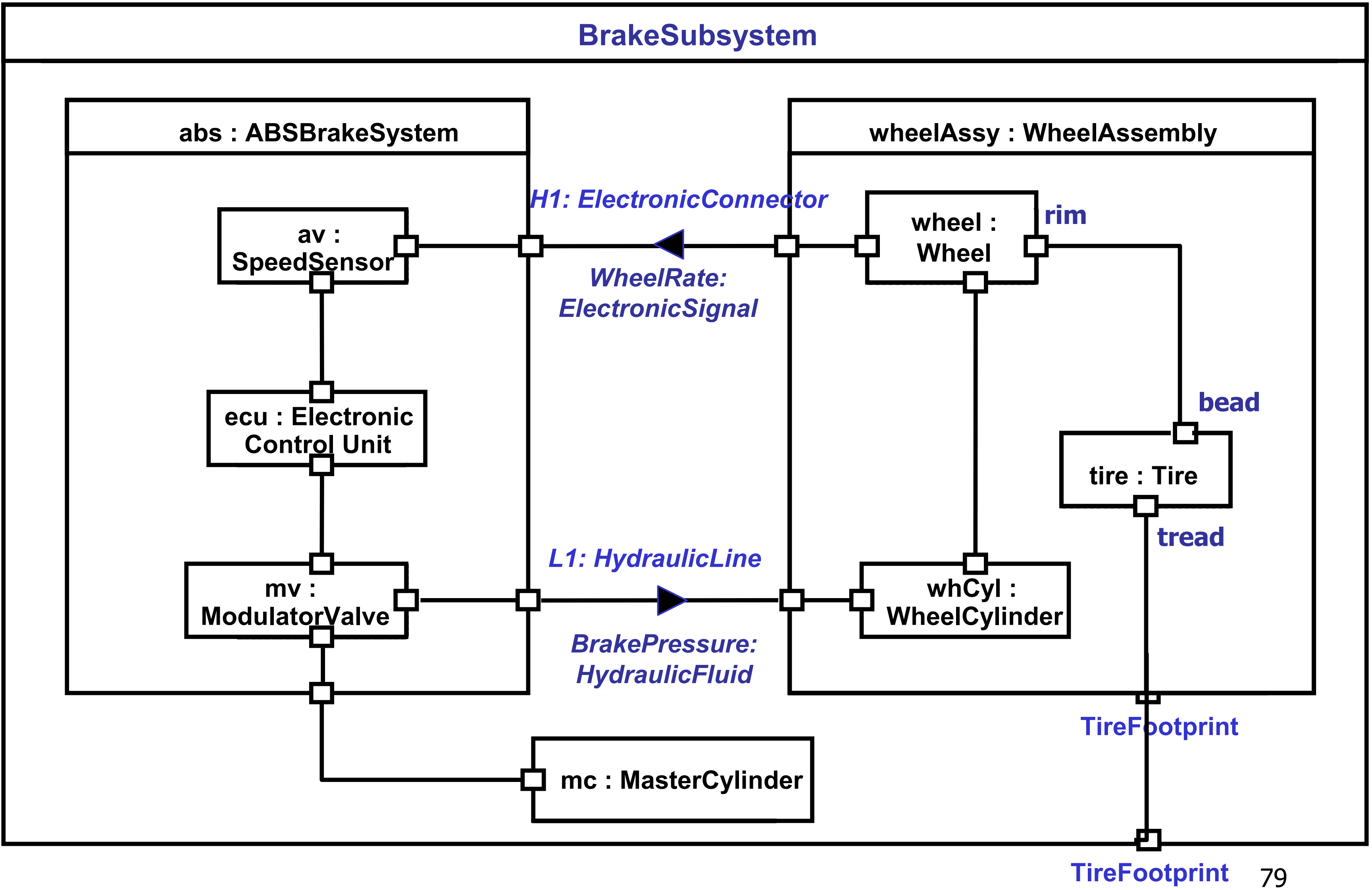
Parts, Ports, Connectors

- Parts are properties that are enclosed by assemblies and typed by classes
  - Additional constraints imposed on SysML part
- Ports are parts in SysML that provide interaction points
  - a port that is attached to a part “p1” is part of the class that types part “p1”
  - notationally represented as a rectangle on the boundary of a part (same as UML 2)
- Connectors bind one part to another
  - can connect parts with or without ports
  - typed by associations
  - structural features of the enclosing class

Assembly Diagram - Example



Item flow on Assembly Diagram



Allocation

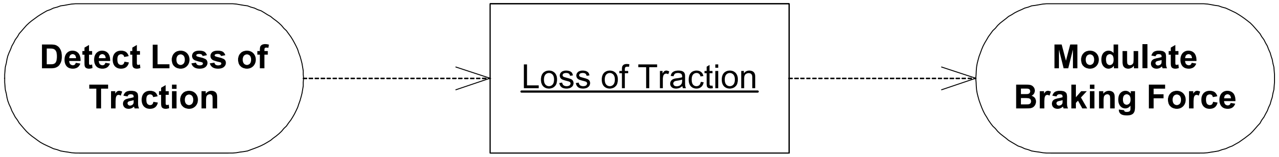
Uses of “Allocation”

Usage	Relationship	From	To
1. Requirement Allocation	UML::trace satisfy	Requirement Packageable Element	Requirement Requirement
2. Behavioral Allocation	allocatedTo	Function (activity), or State Object Node	Assembly/Part Connector
3. Logical Allocation	allocatedTo	Assembly/Part, I/O (logical)	Assembly/Part, I/O (physical)
4. SW deployment onto HW	SysML::deployment	Part (software)	Part (hardware)

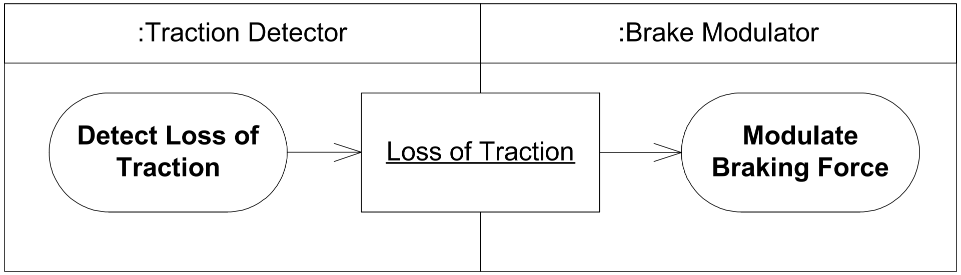
Other types of allocation are intended to be supported

Allocating Behavior to Structure: Example using Swimlanes

- Activity Diagram without Swimlanes:

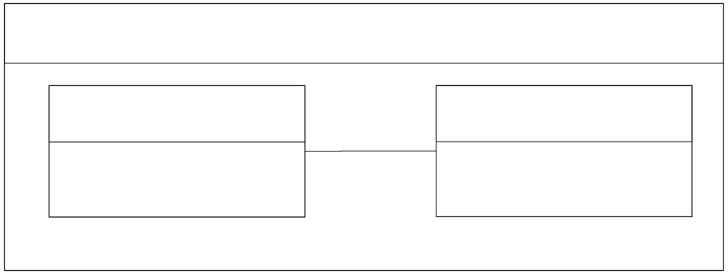


- Activity Diagram with Swimlanes:



SysML Deployment

- More abstract form of deployment than UML 2 that depicts software components deployed to hardware components
- Enables deployment to be depicted on an assembly diagram



SysML™
SYSTEMS MODELING LANGUAGE

Parametric Diagram

UML for SE RFP Requirements

6.5.3.5 Parametric model

UML for SE shall provide the capability to model the following:

a. Properties and their relationships, which represent an arbitrarily complex mathematical or logical expression or constraint, between properties
b. The corresponding mathematical and logical expressions and constraints, which specify the allowable range of values for the properties
c. A reference to the language used to state the expressions and constraints

Note 1: This can include differential equations, logical expressions such as {when Y=7 or X<1}, or other constraints such as {Y< 3x+7}, expressed in a specific language, such as MathML or a programming language.

Note 2: Parametric models are generally captured in analysis models to support feedback and control, performance models, and engineering models for reliability, safety, mass properties, design to cost, etc.

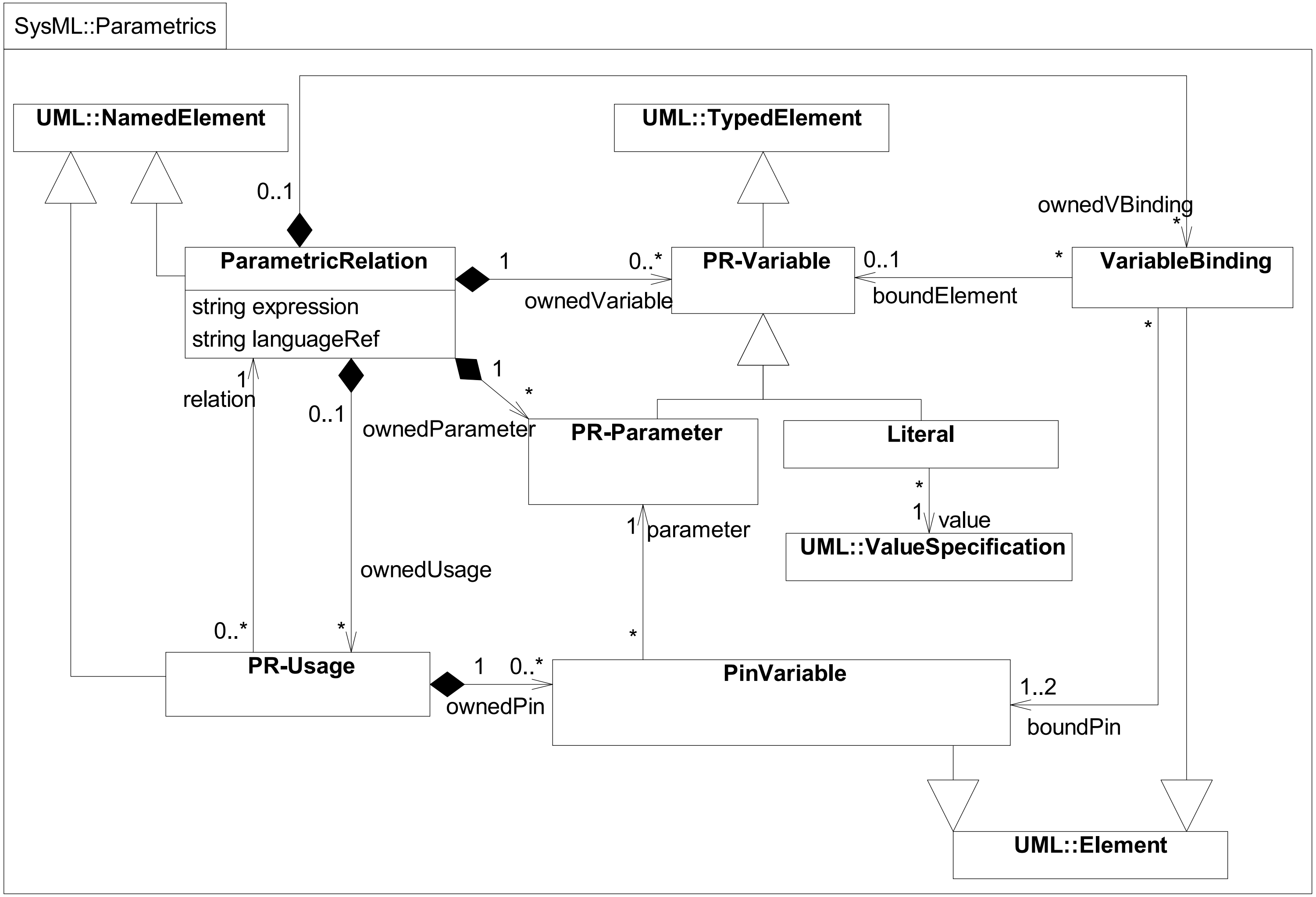
Influences

- Russell Peak (Georgia Tech) – Constrained Objects
  - Georgia Institute of Technology response to the UML for Systems Engineering RFI syseng/02-06-06
- Jacob Axellson (Volvo) – Invariants
  - Volvo Car Corporation response to the UML for Systems Engineering RFI syseng/02-05-06
  - "Model-based Systems Engineering Using a Continuous-Time Extension of UML," Jacob Axellson, INCOSE Journal Volume 5, Number 3 May through June 2002

Parametric Relations

- Used to express relations between quantifiable properties of composite structures
    - Reusable
    - Non-causal (optionally)
- Specification
    - Expression: text string in a …
    - Language (e.g. MathML, OCL …)
    - Parameters identify interface to relation
    - May be defined in terms of other relations
- Usage
    - Used in the context of a SysML assembly
    - Pins bind properties of parts to parameters of relation

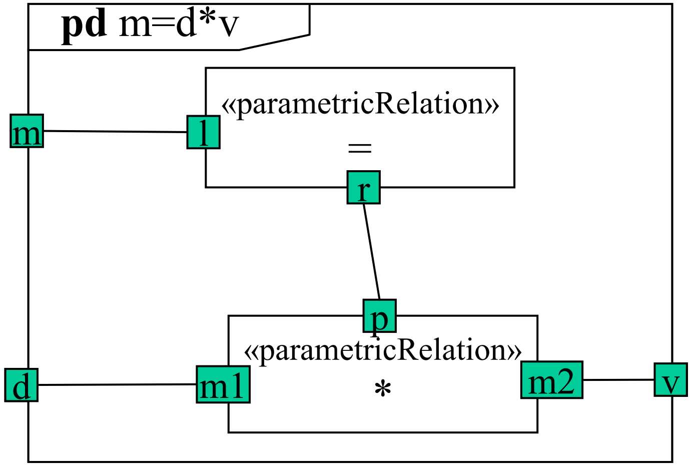
Parametric Metamodel For Specifying Parametric Relationships



Literal example = π

PinVariable (Pin) is a usage of a parameter

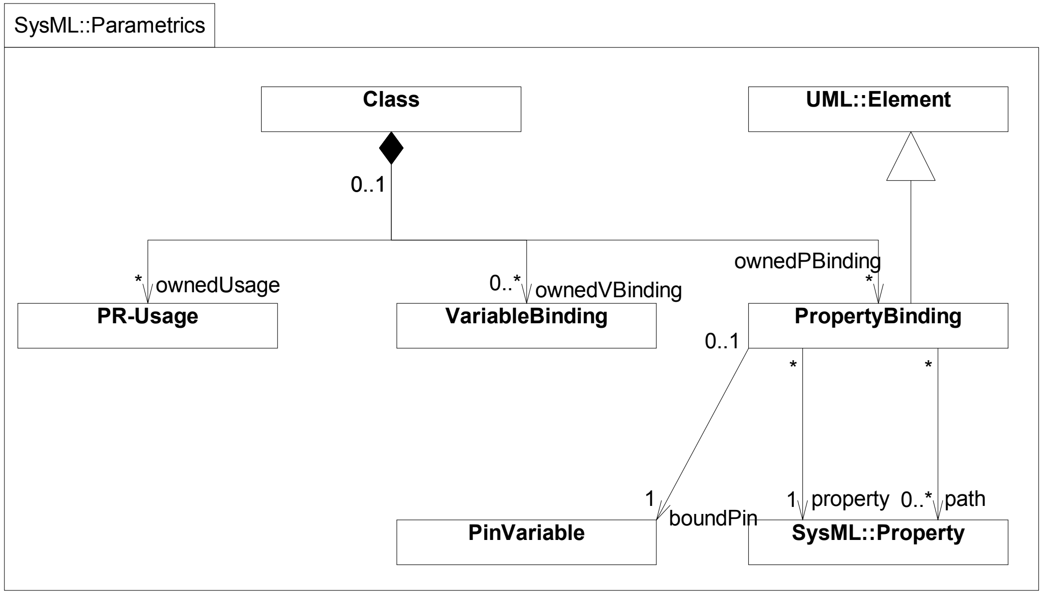
Parametric Relationship - Example



- Frame corresponds to parametric relation
- Pins on frame correspond to parameters
- May use other parametric relations in its specification

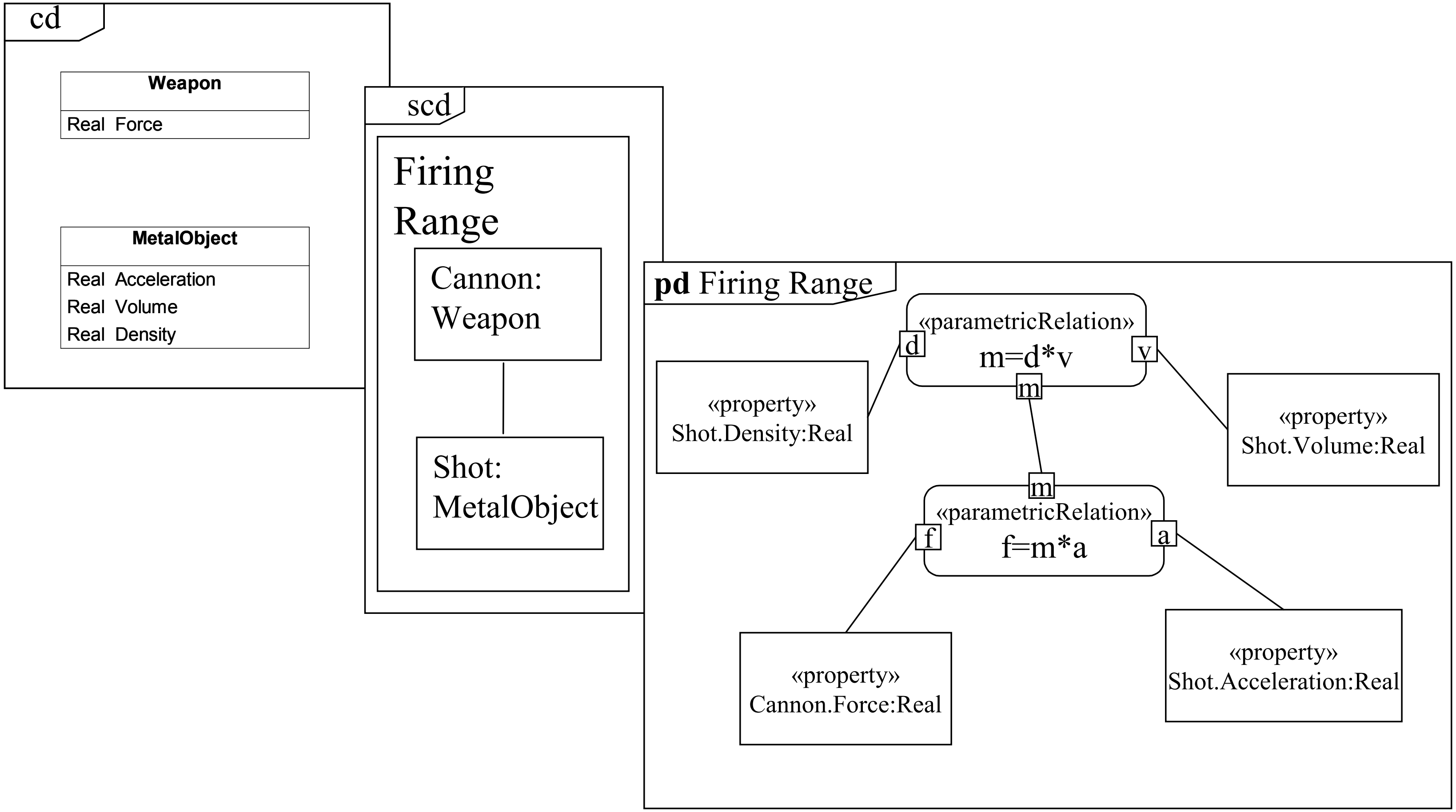
Parametric Metamodel

- Assemblies and classes can use parametric relations to express constraints on their properties and those of their parts.



- Property Bindings bind the pins of the usage to properties

Parametric Example - Usage

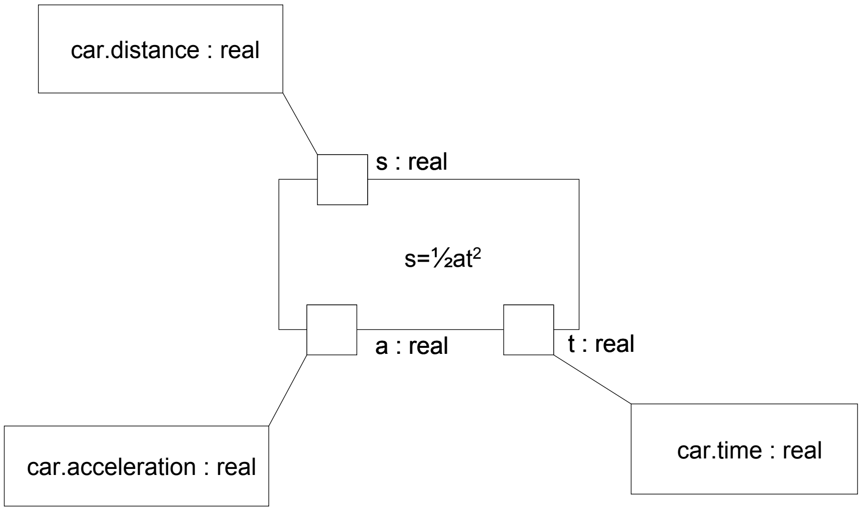


Semantics

- Properties and Literals provide Values, which can be collections (i.e. vector or set of values)
- Pins reference (share) those Values
- Bindings dictate which Values are shared
- Instances of PR-Evaluations are nested according to structure of their Parametric Relations
- SysML relies on external languages (ie. MathML, OCL, ...) for interpreting the parametric relationships (equations)

Treatment of Time

- Ultimately, time is a property of a Continuous and/or Discrete Clock
- Time may be implicit or explicit, depending on need
- Any property may have a time dependence and used in a parametric relationship



Trade-off & Parametrics

- Parametric relation can be used to support evaluation of alternatives (trade-off analysis)
  - Alternatives represented by different models
  - Evaluation function specified as a parametric relationship in terms of:
    - Criteria, weighting
    - Probability distributions can be applied to properties
    - Evaluation result can be viewed as a measure of effectiveness
  - Can be represented in typical table format
- Approach will be formalized post V1.0

Alternative Approaches for Parametrics

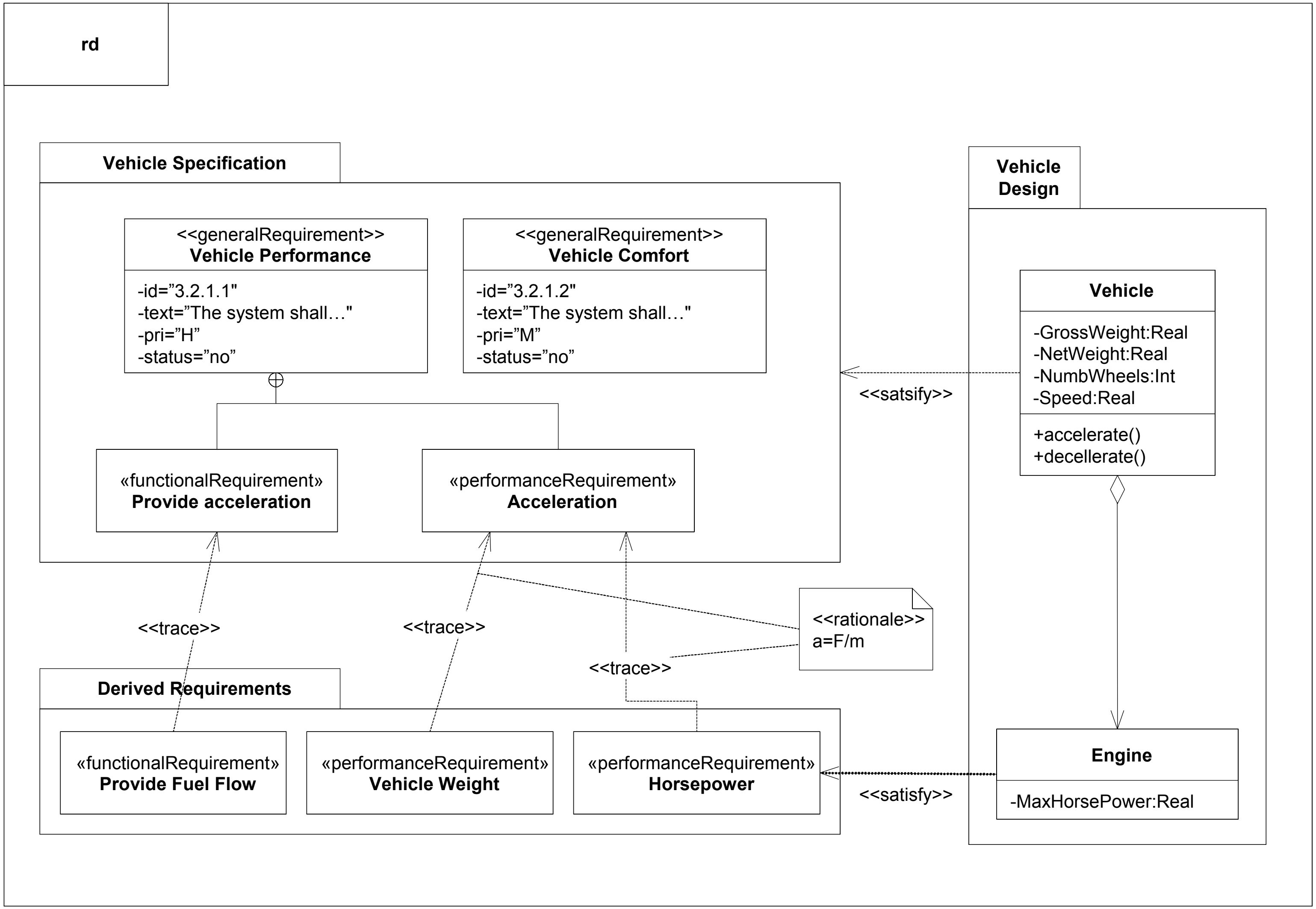
- Use of Activity Model
  - Parametric relationships have no causality – i.e. parameters may have no direction, unlike activities
  - Parametrics have a much simpler model (and semantics) than UML activities
- Use of OCL
  - OCL has no obvious analog for parametric relation
    - Query assumes return value, can't have unspecified parameter direction
  - OCL has no concept of property binding
  - Parametric has much simpler semantics than OCL
- Currently examining a metamodel change to depict a parametric relationship as a type of Constraint

Requirements Diagram

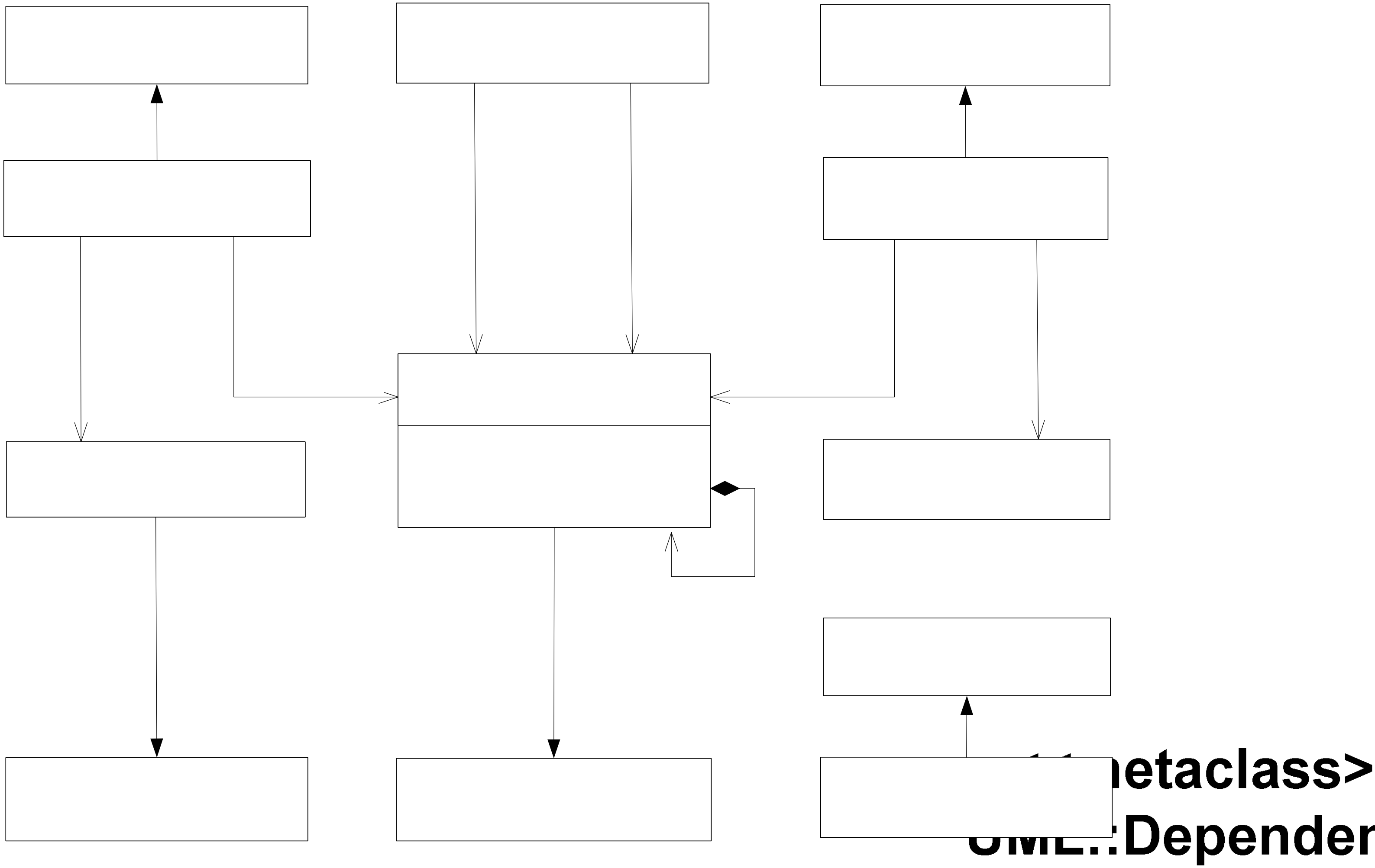
Approach

- The Requirements diagram can represent the following:
  - text based requirements and properties (e.g., id, text statement, criticality, etc)
  - package as a set of requirements
  - requirements decomposition into constituent elements
  - traceability between derived and source requirements
  - design elements satisifying one or more requirements
  - verification of a requirement by a test case
  - rationale for requirements traceability, satsifaction, etc
- Alternative graphical, tabular and tree representations

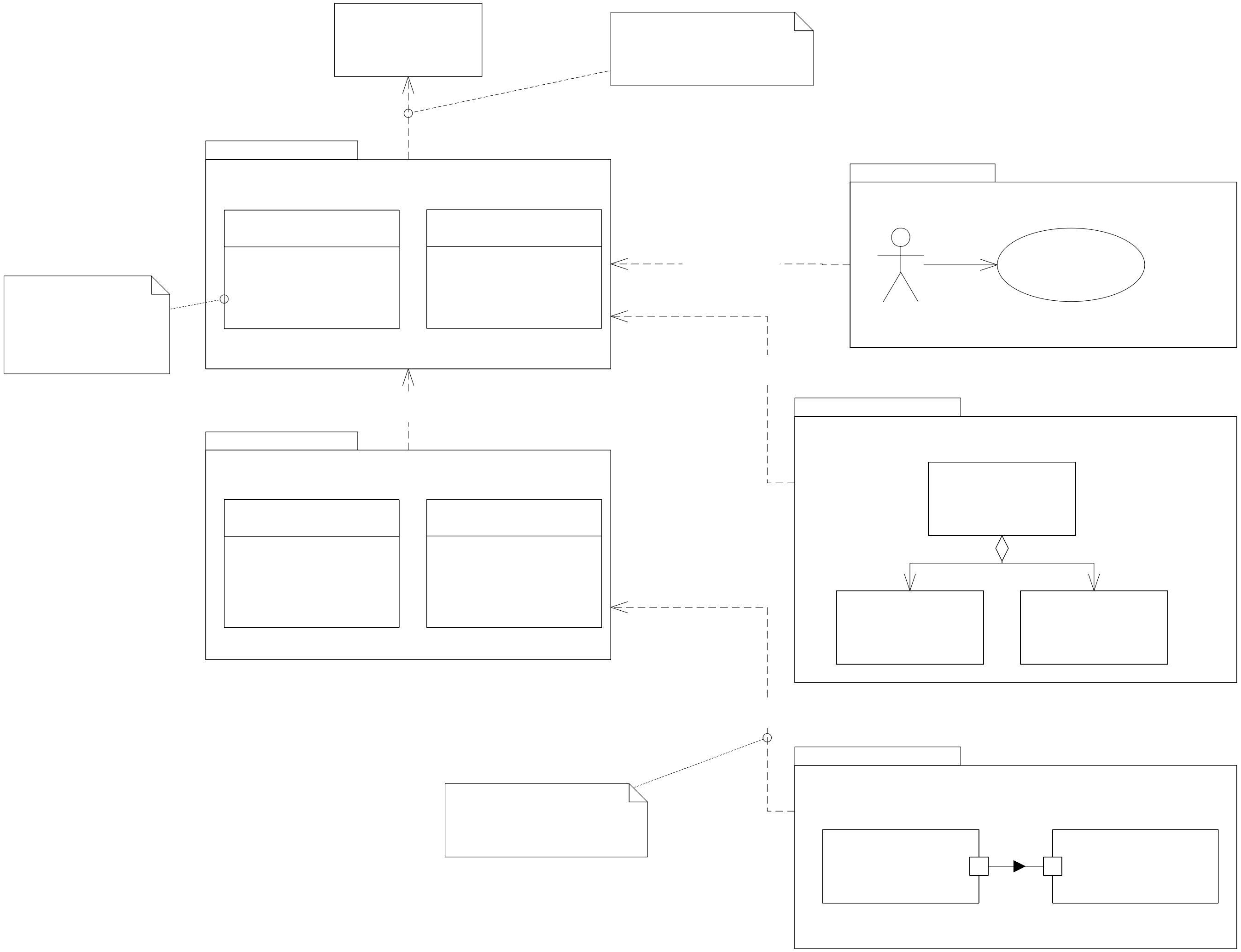
Requirements Diagram Example



Requirements Metamodel



Requirements Flowdown

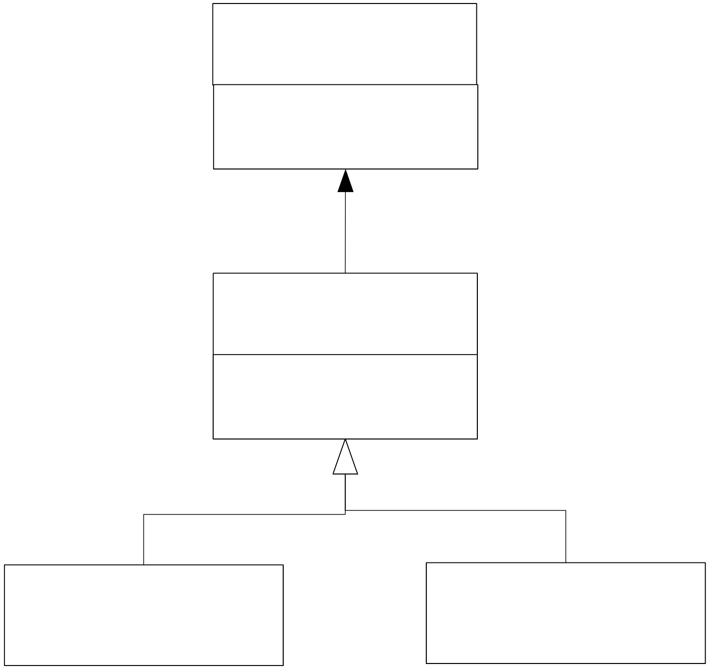


Requirement Classification

- Basic attributes in Requirement
  - text
  - ID
  - criticality
- Users create stereotypes for specific types of requirements, e.g. performance
  - add properties
  - add associations (e.g., parametric relations)
- Predefined profiles for standard requirement types

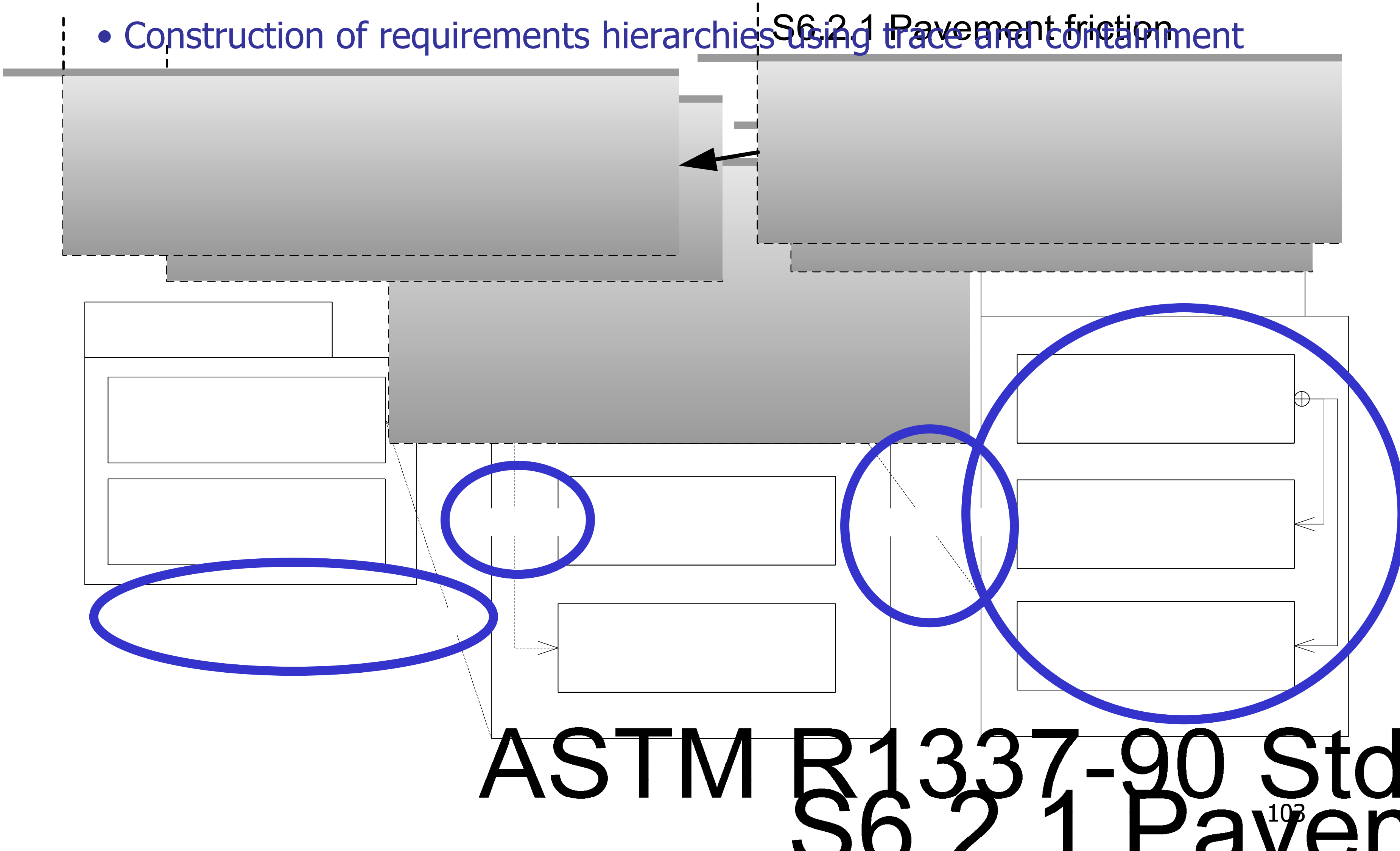
Requirements Stereotypes

- Accommodates user specified requirements subclasses



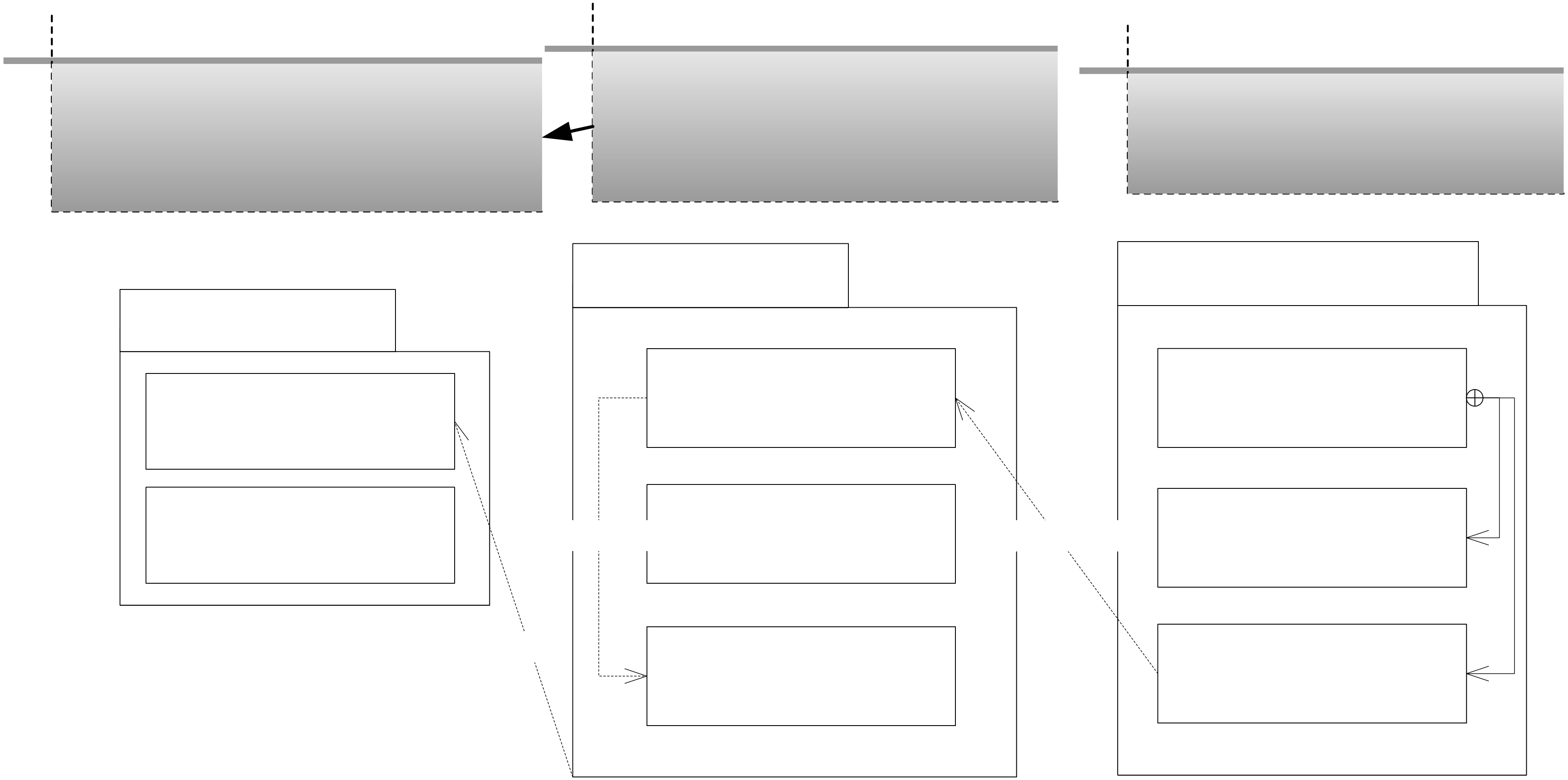
Basic semantic

- Construction of requirements hierarchies using trace and containment



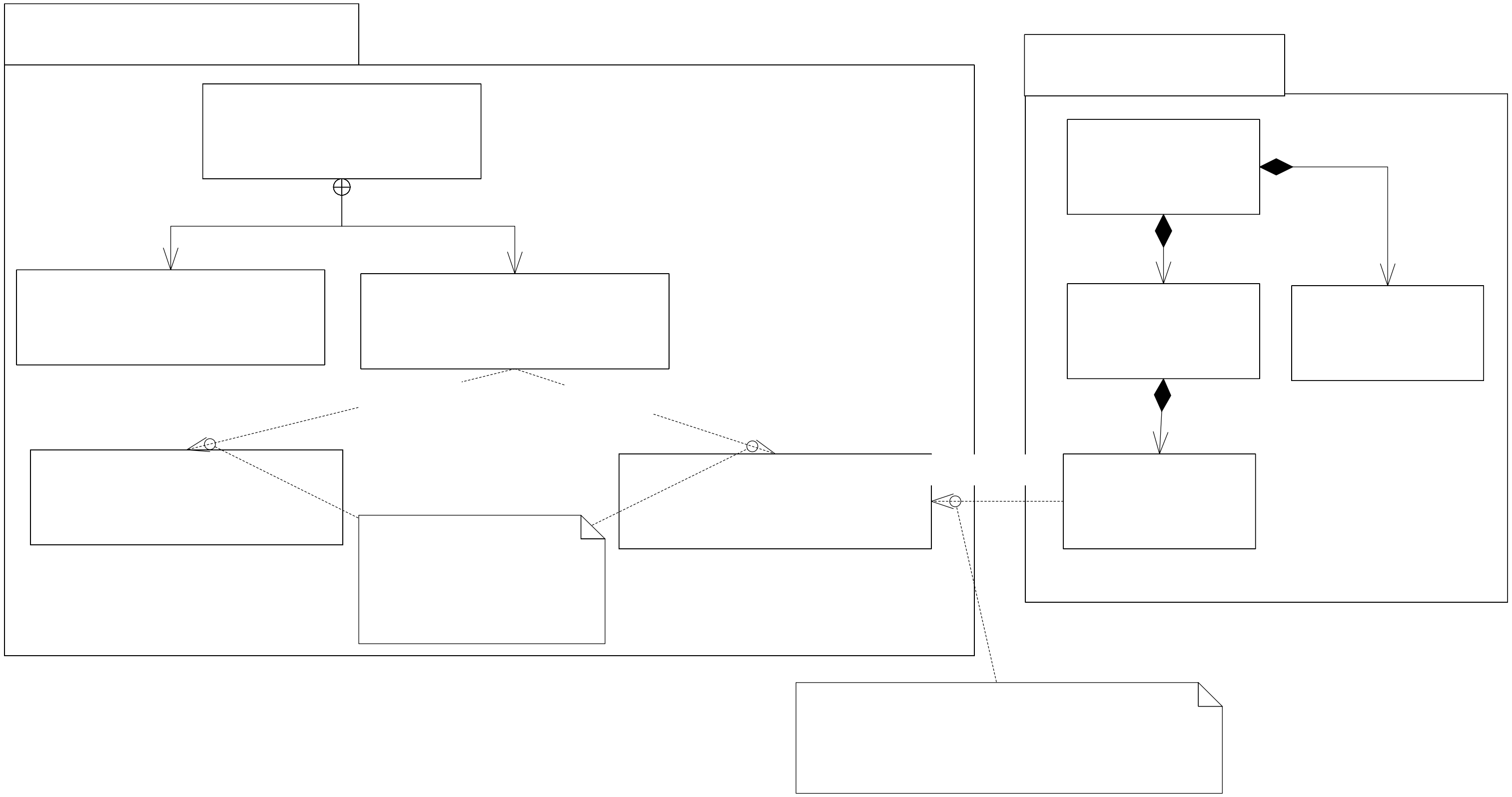
Requirements Hierarchies

- Construction of requirements hierarchies using trace and containment



Requirements & Design Rational

- Capture rational for trace and assign relationships



AP233 Alignment

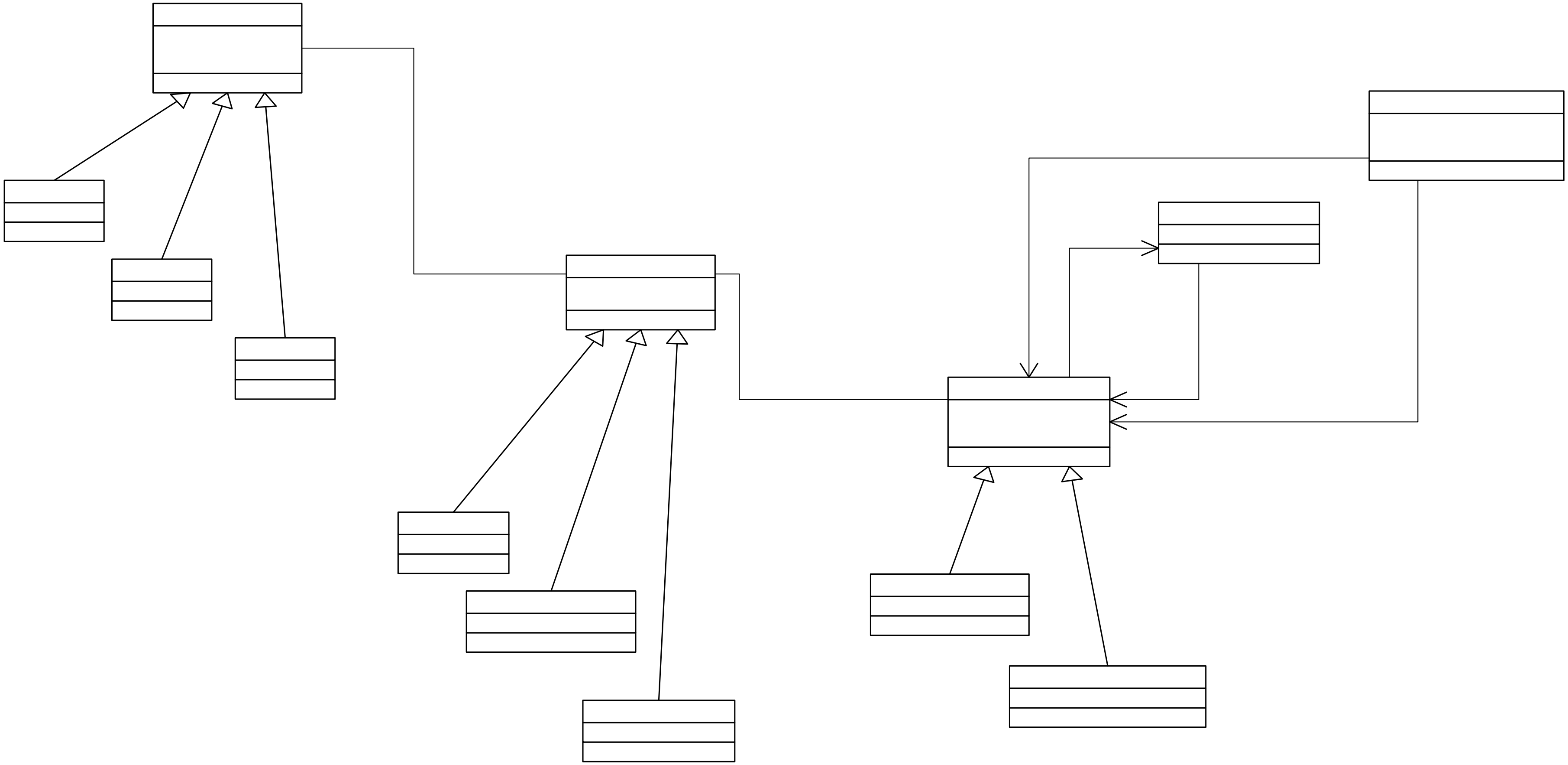
Background

- What is AP233
  - a STEP data modeling project that is providing a series of systems engineering data models that map into existing families of systems engineering tools and sub-domains

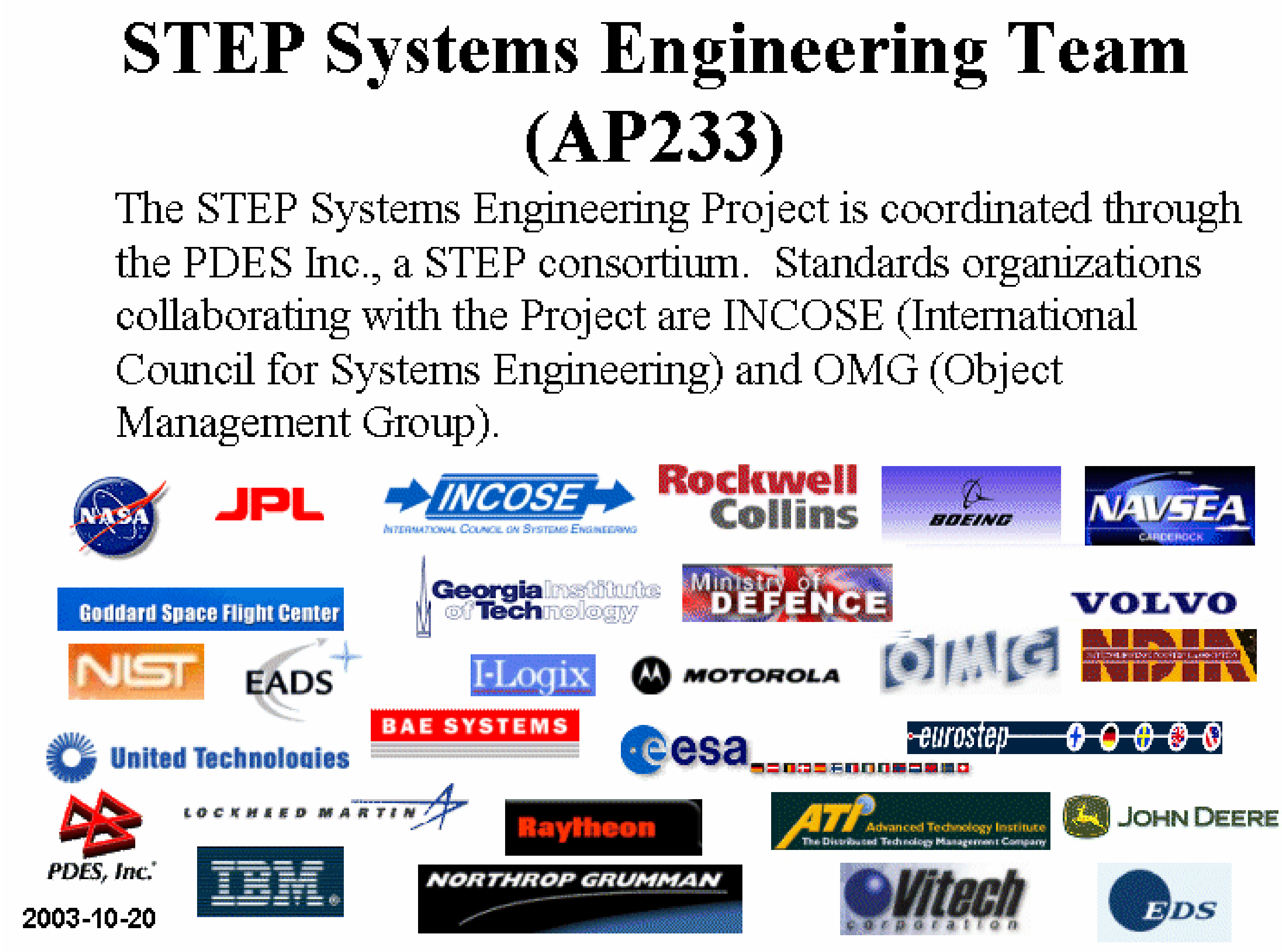
AP233 Module Sets

- **Requirements**
- Structural Models
- Behavioral Models
- Risk Analysis
- Rules
- Validation and Verification
- Security
- Scheduling
- WBS
- Cost Models
- Organizational Structure
- PDM extensions
- AP Interface Modules
- Data Representation

AP233/STEP-PDM Schema Re-use



AP233 Team



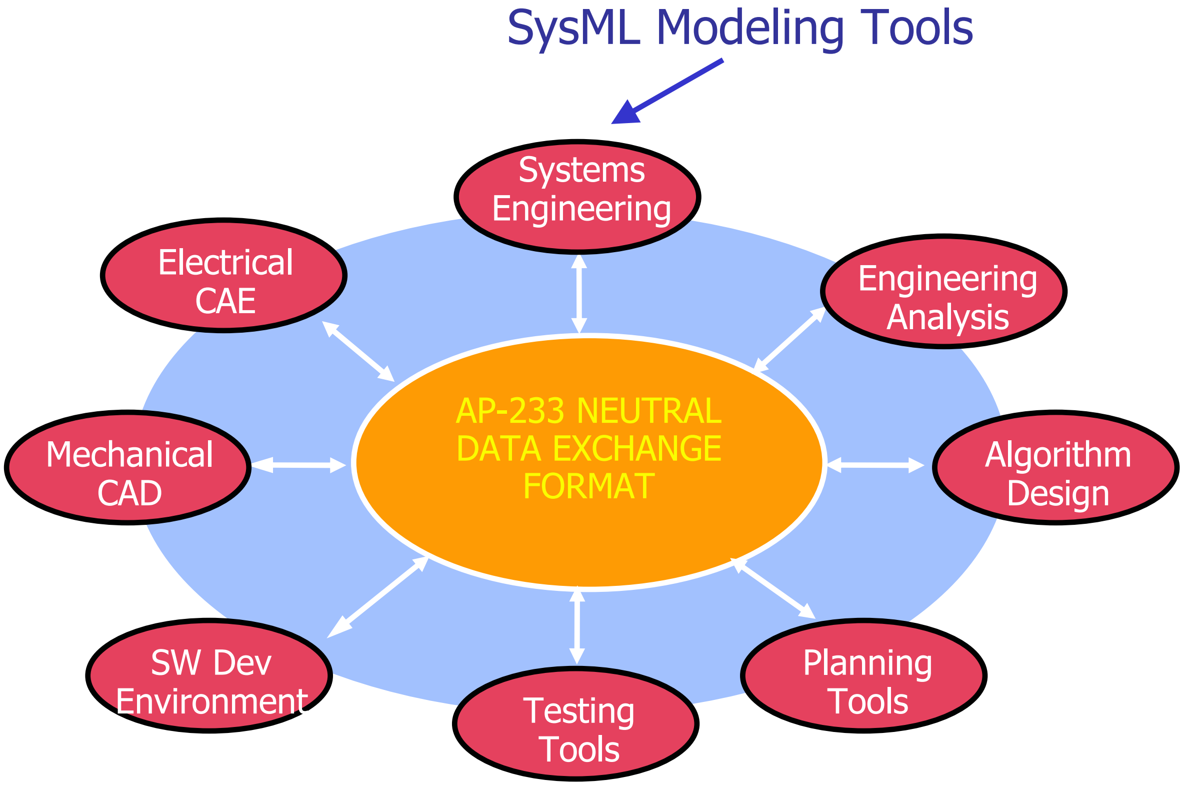
STEP Systems Engineering Team (AP233)

The STEP Systems Engineering Project is coordinated through the PDES Inc., a STEP consortium. Standards organizations collaborating with the Project are INCOSE (International Council for Systems Engineering) and OMG (Object Management Group).

NASA
JPL
INCOSE
International Council on Systems Engineering
Rockwell Collins
BOEING
NAVSEA
CARDEROCK
Goddard Space Flight Center
Georgia Institute of Technology
Ministry of DEFENCE
VOLVO
NIST
EADS
I-Logix
MOTOROLA
OMG
NDIA
BAE SYSTEMS
United Technologies
esa
eurostep
PDES, Inc.
LOCKHEED MARTIN
Raytheon
ATI Advanced Technology Institute
The Distributed Technology Management Company
JOHN DEERE
IBM
NORTHROP GRUMMAN
Vitech corporation
EDS

2003-10-20

SysML and AP-233 Alignment

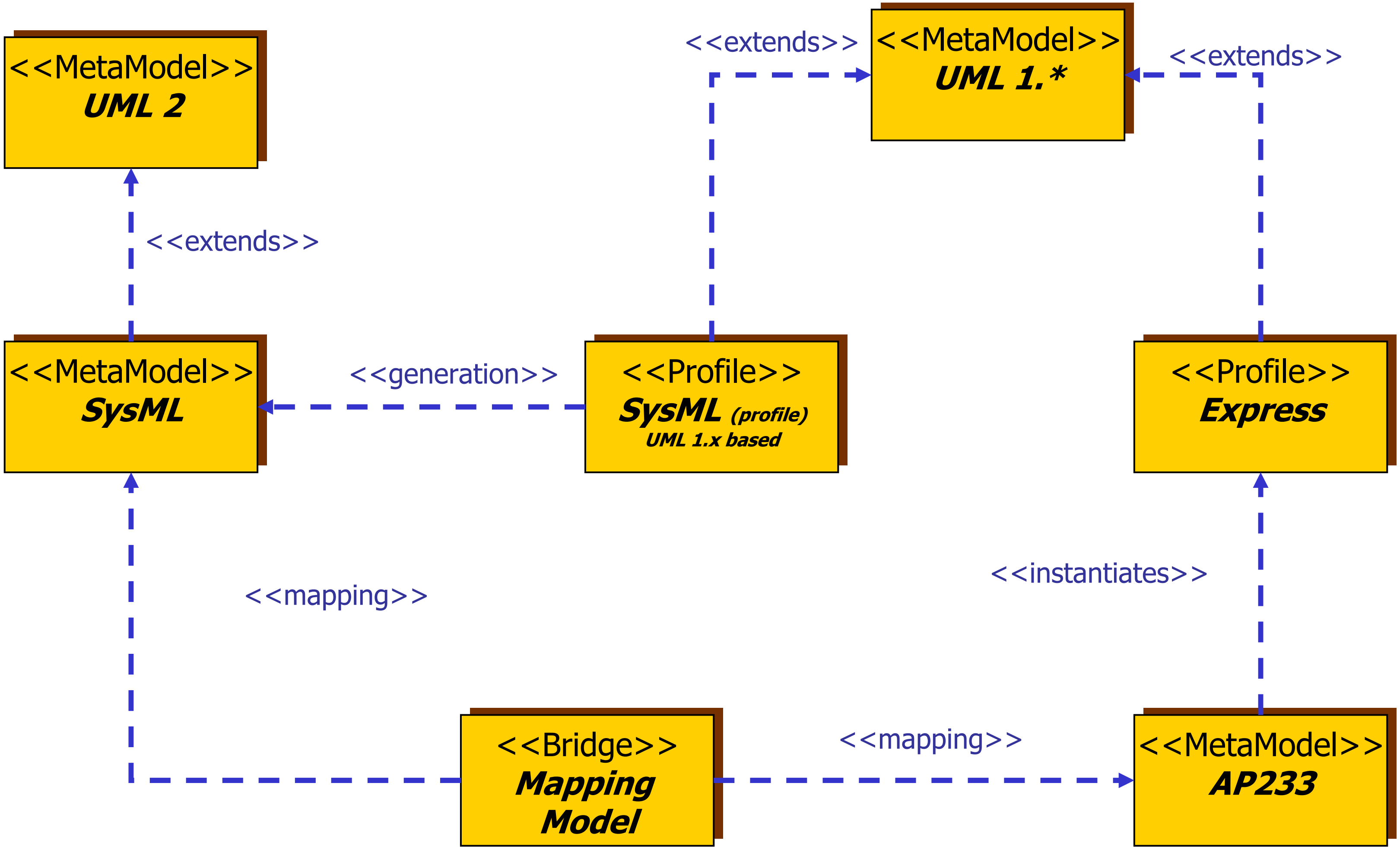


Alignment Objective

- Align SysML and AP233 models
- Provide meta-model mapping
- Provisions for an independent public domain SysML/AP233 API
- Set-up of data-exchange test-bed

Alignment Approach

SYSTEMS MODELING LANGUAGE



<<MetaModel>>
UML 2

<<extends>>

<<MetaModel>>
SysML

<<generation>>

<<Profile>>
SysML (profile)
UML 1.x based

<<extends>>

<<MetaModel>>
UML 1.*

<<extends>>

<<Profile>>
Express

<<instantiates>>

<<mapping>>

<<Bridge>>
Mapping Model

<<mapping>>

<<MetaModel>>
AP233

Wrap Up

Wrap Up

- SE DSIG established as joint INCOSE/OMG initiative to extend UML to support SE and align with AP-233
    - kickoff in September 2001
    - joint SE DSIG/INCOSE/AP-233 requirements analysis and review
    - issued UML for SE RFP in March 2003
- SysML Partners established in May 2004 to respond to RFP
    - includes wide range of contributors from industry, tool vendors and government agencies
- SysML approach architecturally extends UML 2 Superstructure
    - extensively reuses a subset of UML 2 “out of the box”
- Major changes to UML 2 include:
    - enhancements to composite structure and activity diagrams
    - two new diagram types (requirements and parametrics)
    - other changes include allocation relationships and auxiliary constructs
    - SysML alignment with ISO AP-233
- Working towards adoption of SysML v1.0 in H2 2004
- Technical approach is being validated by prototypes and partial implementations

References

- UML for SE RFP
  - OMG doc# ad/03-03-41
- [UML2 2003] UML 2 Superstructure (Final Adopted Specification)
  - OMG doc# ptc/03-08-02
- [Bock 2003-4] Activities
  - INCOSE Journal
    - www3.interscience.wiley.com/cgi-bin/abstract/106557123/ABSTRACT
  - Journal of Object Technology
    - www.jot.fm/issues/issue_2003_07/column3
    - www.jot.fm/issues/issue_2003_09/column4
    - www.jot.fm/issues/issue_2003_11/column1
    - www.jot.fm/issues/issue_2004_01/column3
- [Kobryn 2003] C. Kobryn and E. Eric Samuelsson, "Driving Architectures with UML 2.0", Telelogic White Paper, Aug. 2003.
- INCOSE Insight Articles (to be published)

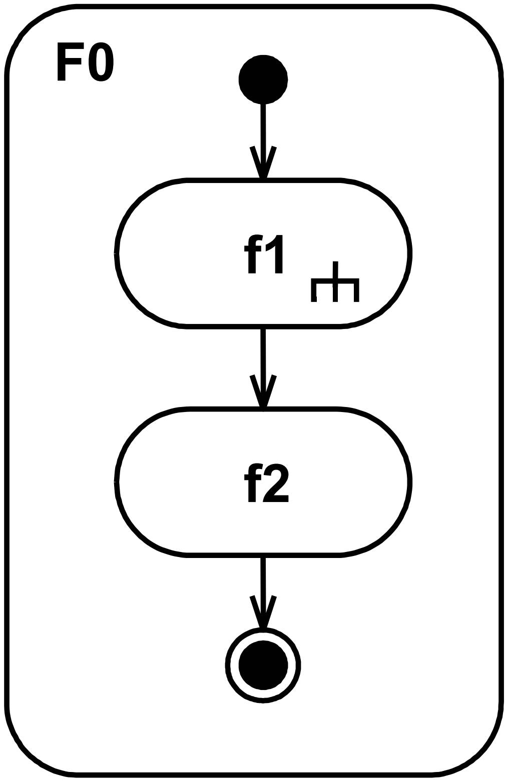
Further Info

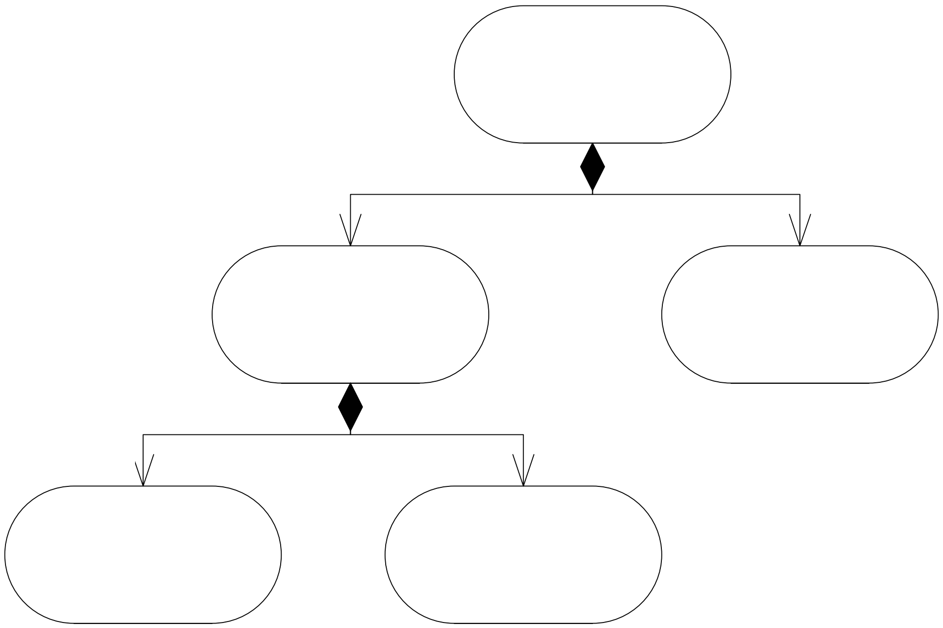
- Web
  - www.sysml.org
- Chairs
  - Cris Kobryn
    - cris.kobryn@telelogic.com; cris@sysml.org
  - Sandy Friedenthal
    - sanford.friedenthal@lmco.com; sandy@sysml.org

Backup

Functional (Activity) Decomposition

Actions are usage of activities and follows usage pattern similar to assembly/part





Activity

Functional Decomposition