



SysML v. 1.0a Specification

(revised OMG Submission)

SysML Partners

www.sysml.org

December 2005

Overview

- Proposal status
- Change summary
- Statement of Compliance & Requirements Traceability
- Verification & Validation
- Proof of Concept:
Executable Sample Problem
- Evaluation and finalization process
- Next steps



Background

What is SysML?



- **Systems Modeling Language (SysML)**
 - SysML is a domain-specific visual modeling language for systems engineering.
 - SysML extends, and is compatible with, the Unified Modeling Language (UML) for software engineering.
 - SysML supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities.
 - SysML is being developed by an open source project, which is organized and supported by the SysML Partners, an informal association of industry leaders and tool vendors.

SysML Partners



- SysML Core Partners (founders and main contributors)
 - Gentleware¹
 - Motorola¹
 - Northrop Grumman
 - PivotPoint Technology
 - Telelogic¹

- Other contributors
 - American Systems Corporation
 - ARTISAN Software Tools
 - BAE SYSTEMS
 - The Boeing Company
 - Ceira Technologies
 - Deere & Company
 - EADS Astrium GmbH
 - EmbeddedPlus Engineering
 - Eurostep Group AB
 - I-Logix, Inc.
 - IBM
 - INCOSE
 - Israel Aircraft Industries
 - Lockheed Martin Corporation
 - oose.de
 - Raytheon Company
 - THALES
 - Georgia Tech
 - NASA

¹Submitted OMG Letter of Intent

SysML Milestones



- UML for SE RFP issued – 28 March 2003
- SysML Partners' kickoff meeting – 6 May 2003
- SysML v. 0.3 – OMG initial submission – 12 Jan. 2004
- INCOSE Review – 25-26 Jan. 2004
- INCOSE Review – 25 May 2004
- SysML v. 0.8 – OMG revised submission – 2 Aug. 2004
- SysML v. 0.85 – OMG revised submission – 11 Oct. 2004
- SysML v. 0.9 – OMG revised submission – 10 Jan. 2005
- INCOSE Review – 29-30 Jan. 2005
- SysML v. 0.9 Addendum – OMG submission update – 30 May 2005
- INCOSE Review – 10 July 2005
- SysML v. 1.0a – OMG revised submission – 14 Nov. 2005
- INCOSE Review – Dec. 2005 – Jan. 2006
- SysML v. 1.0 GA – Q1 2006

Top-Level RFP Requirements



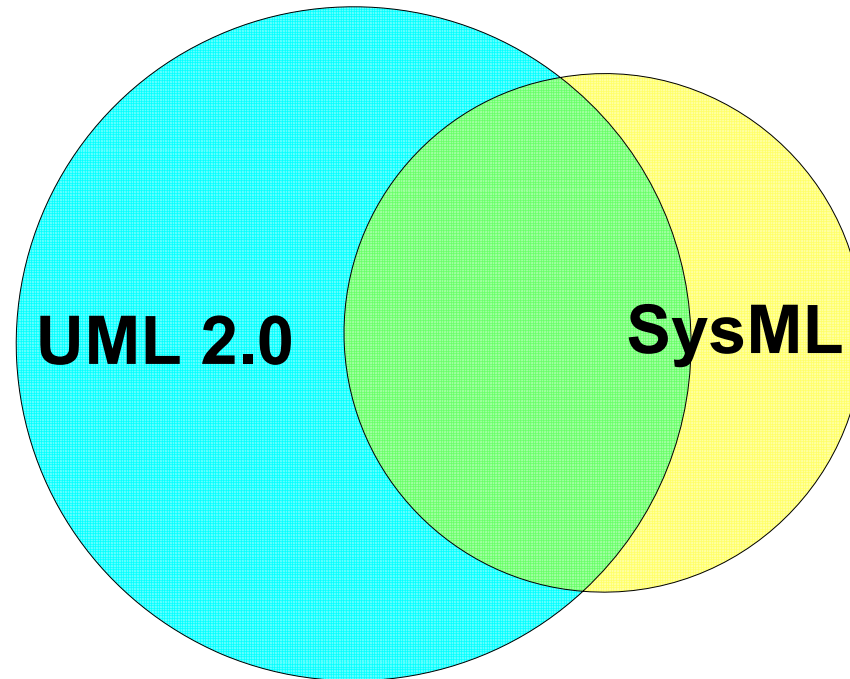
- Structure
 - e.g., system hierarchy, interconnection
- Behavior
 - e.g., function-based behavior, state-based behavior
- Properties
 - e.g., parametric models, time property
- Requirements
 - e.g., requirements hierarchy, traceability
- Verification
 - e.g., test cases, verification results
- Other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

Standard UML 2.0 Support for SE



- Standard UML 2.0 satisfies most of the needs of SE today!
 - Structural decomposition and interconnection
 - via Parts, Ports, Connectors
 - Behavior decomposition
 - e.g., Sequences, Activities, State Machines
 - Enhancements to Activity diagrams
 - e.g., data and control flow constructs, activity partitions/swim lanes
 - Enhancements to Interaction diagrams
 - e.g., alternative sequences, reference sequences

UML 2 Reuse



 **Common diagrams: Activity, Block Definition (UML2::Class), Internal Block (UML2::Composite Structure), Package, Sequence, State Machine, Use Case**

 **New diagrams: Parametric Constraint, Requirement**

*Created by Systems Engineers **FOR** Systems Engineers!*

SysML Design Approach



- Reuse and extend UML 2.0
 - select subset of UML 2.0 that is reusable for SE applications as baseline (remove gratuitous SW constructs and diagrams)
 - add new constructs and diagrams needed for SE
 - SysML = UML2⁺⁺⁺
- AP-233 alignment
 - align with evolving AP-233 SE Data Interchange Standard
- **Incremental development**
 - extend the language incrementally, using SE feedback to ensure new extensions are valid

Fundamental Design Principles

- Parsimony
- Reuse
- Modularity
- Layering
- Partitioning
- Extensibility
- Interoperability

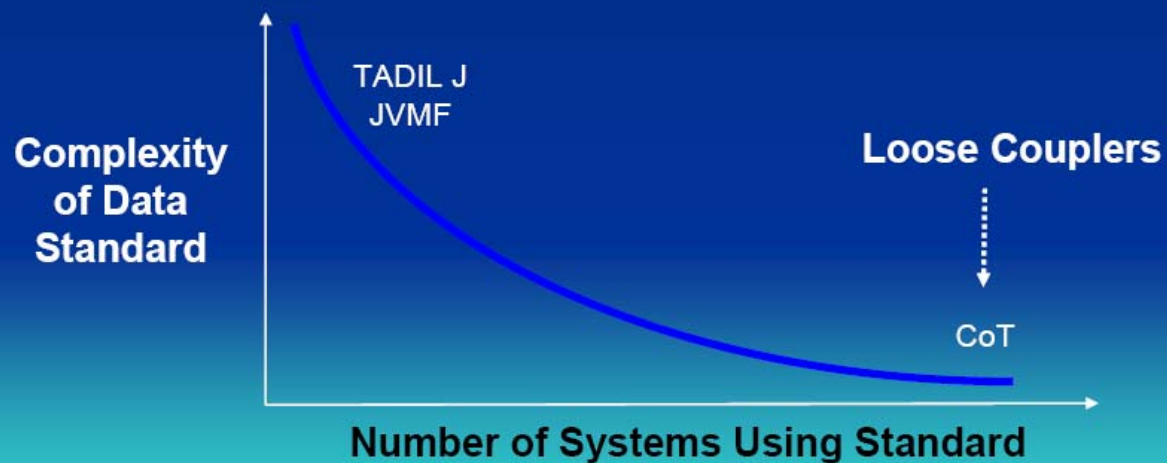
Parsimony Revisited: Does size matter?

- Power Law in Data Standards
 - R. Byrne, "Managing Complexity: An Approach to Net-Centric Ops," presentation, MITRE Corporation, September 2005.
- MDA
 - Dave Thomas, "UML – Unified or Universal Modeling Language?: UML2, OCL, MOF, EDOC - The Emperor Has Too Many Clothes," Journal of Object Technology, Vol. 2, No. 1, 2002.
 - Experience with UML 1.x vs. UML 2.0
 - Experience with CDIF vs. MOF 1.x and MOF 2.0

Power Law in Data Standards

Power Law in Data Standards

- The more systems that must adopt a common standard, the simpler it must be

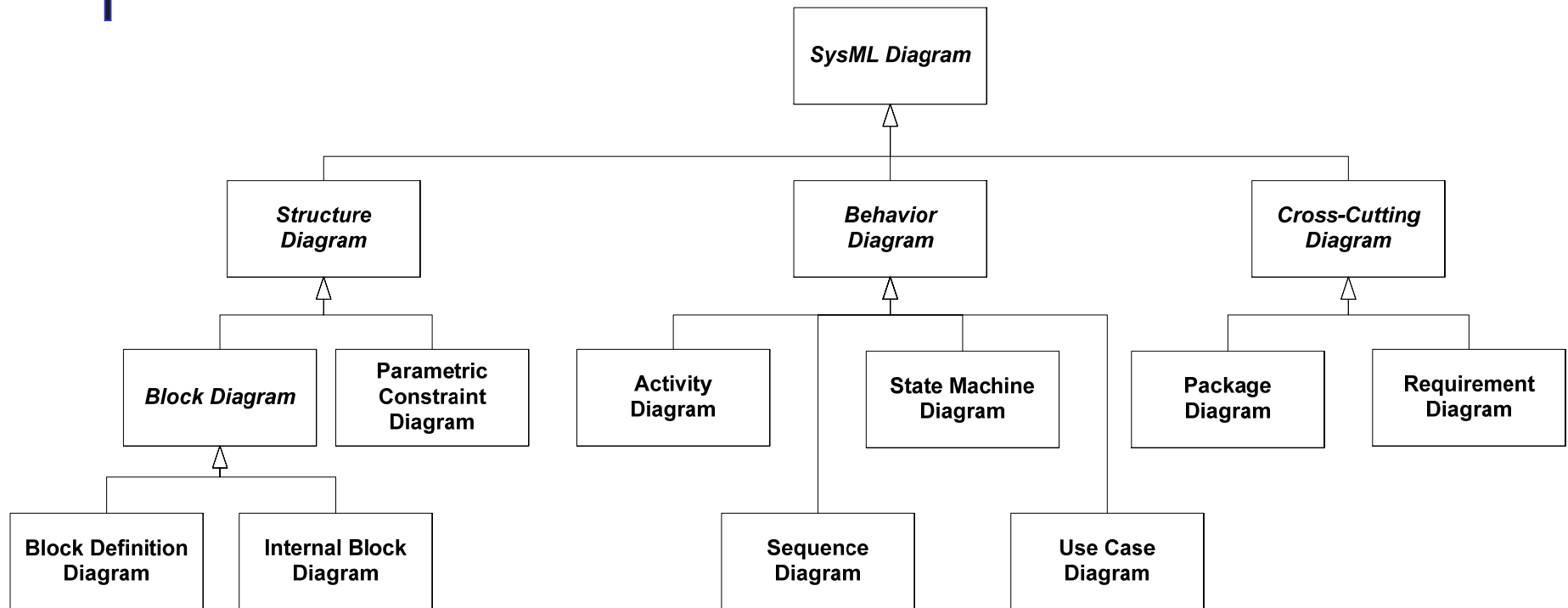


[Byrne 2005]

SysML Diagrams

- Reuses UML 2 diagrams for:
 - Use Case diagram
 - State Machine diagram
 - Interaction diagrams (Sequence)
- Extends the following diagrams
 - Block Definition Diagram (based on UML Class diagram)
 - Internal Block diagram (based on UML Composite Structure diagram)
 - Activity diagram (based on UML Activity diagram)
- Adds the following diagrams
 - Parametric Constraint diagram
 - Requirement diagram

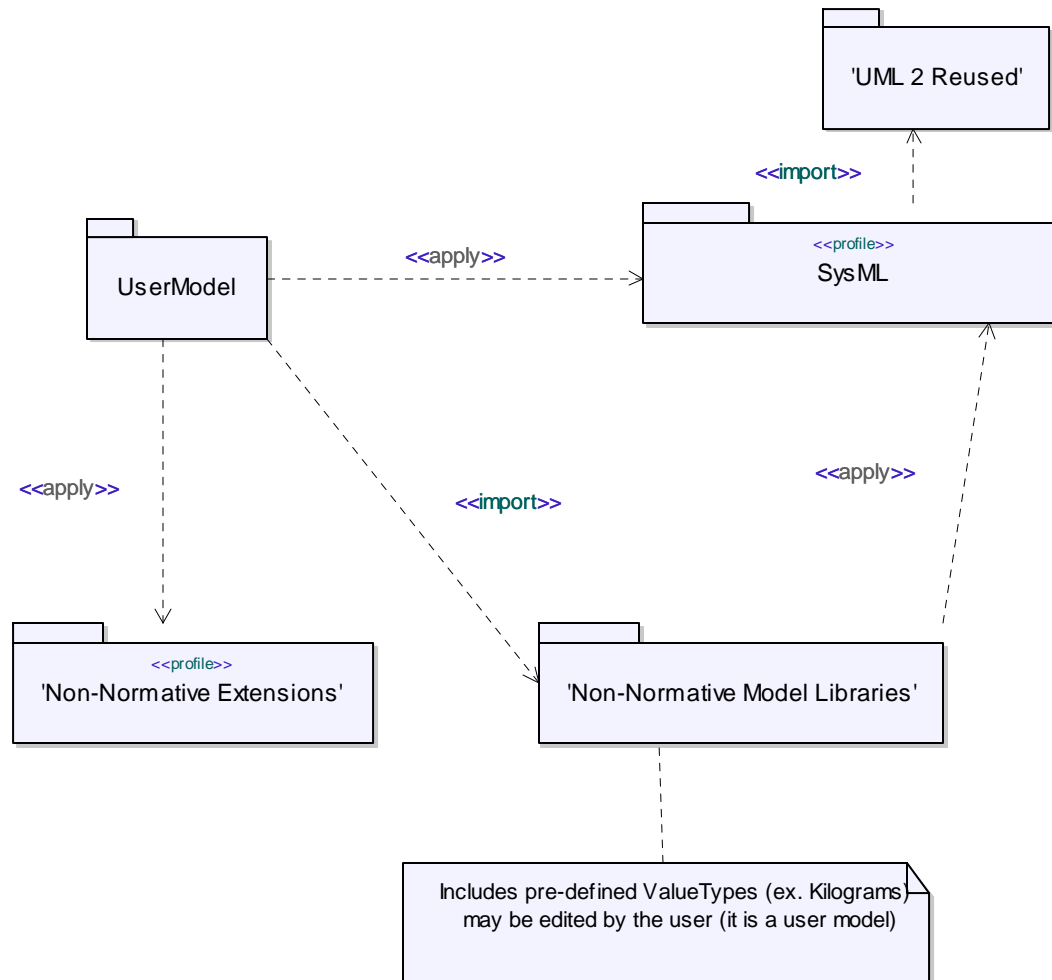
SysML Diagram Taxonomy



Language Architecture

Language Architecture

<<informal>>package 'SysML Meta-Model' {1/1}





Proposal Status

Specification Status



- SysML Specification v. 1.0a completed
 - submitted to OMG (OMG doc# ad/05-11-05)
 - support document: Statement of Compliance & Requirements Traceability Matrix (OMG doc# ad/05-11-06)
- Other support documents available on SysML public web
 - abstract syntax in HTML and XML tool-specific format
 - XMI v. 1.1 available; XMI v. 2.1 in progress
 - executable sample problem available (see demo)

Issue Status



- 244 SysML v. 0.9 issues submitted to SysML web feedback or public mailing list
 - 221 issues have been resolved
 - 23 issues "In Progress"
 - will be resolved in SysML v. 1.0

Follow Up on INCOSE IW MDSD WG Recommendations



- Improve SysML tutorial
 - emphasize 5 Core diagrams and be driven by Requirements diagrams
 - replace UML-specific definitions with domain-specific explanations
 - present update at INCOSE Symposium (MDSD plenary)
 - **Actions:** Appendix B Sample Problem improved in coverage, complexity and explanations so that it can be easily reused for a tutorial.
- Increase readability of SysML specification for engineers and tool vendors
 - replace UML-specific definitions with domain-specific explanations
 - include a domain metamodel
 - **Actions:** Specification rewritten to improve readability for SE's. Updated Glossary will be made available prior to INCOSE MDSD review.

Follow Up on INCOSE IW

MDSW WG Recommendations (cont'd)



- Include a model library for Requirement taxonomy
 - include MeasureOfEffectiveness (MOE; properties: weight, optimizationDirection)
 - MOE may also include a complementary Parametric construct to effect MOE constraints
 - **Actions:** Extensible requirements taxonomy that doesn't break interoperability incorporated as part of the normative specification. MOE's supported as non-normative extensions.
- Include a model library for Assemblies that includes PhysicalAssembly (properties: supplier, modelNumber, serialNumber, lotNumber)
 - **Actions:** Non-normative extensions to capture hardware (HWCI) and software configuration item (CSCI) identification included in SysML v. 1.0b

Follow Up INCOSE IW

MDSD WG Recommendations (cont'd)



- Harmonize concepts, constructs, and usage examples for Allocations
 - make implicit Allocations explicit
 - test usability of multiple UI options via vendor prototypes
 - **Actions:** Refined explicit allocations and tested usability via prototyping.
- Encourage and promote vendor SysML prototypes at INCOSE Symposium vendor exhibits
 - **Actions:** Presented prototypes at July 2005 INCOSE symposium on show floor and to MDSD/AP233 working groups. SysML V1.0a supported by TAU G2 2.6 released November 2005.

Evaluation Status

- Two external reviews are in progress
 - INCOSE MDSD WG/ISO AP233 WG trade study
 - ADTF evaluation team
- Partners will actively support both reviews and will incorporate recommendations to improve standard

SysML Specification Outline Update



- Preface
- TOC, List of Figures, List of Tables
- Part I - Introduction
- Part II – Structural Constructs
 - Blocks
 - Parametric Constraints
- Part III – Behavioral Constructs
 - Activities
 - Sequences
 - State Machines
 - Use Cases
- Part IV – Crosscutting Constructs
 - Requirements
 - Allocations
 - Model Management
 - Types
 - Auxiliary Constructs
 - Profiles & Model Libraries
- Appendices
 - Diagrams
 - Sample Problem
 - Non-Normative Extensions
 - Non-Normative Model Library
 - OMG XMI Model Interchange
 - ISO AP233 Model Interchange
 - Requirements Traceability Matrix
 - Index

Change Summary

Top-Level Changes

- "Less is more": Reduced complexity increases semantic expressiveness.
 - Strict profile approach that extends a small subset of UML increases potential for interoperability
 - Many UML constructs non-meaningful to systems engineers have been removed to make language easier to learn and apply
 - Numerous SysML v. 0.9 constructs that are unproven, problematic to implement have been removed to reduce implementation risk
- Sample problem reworked
 - Complete, consistent, complex problem validates language expressiveness and pragmatism
 - Provides a unifying thread throughout the specification to illustrate concepts
 - Based on an executable model (compare disconnected Visio drawings) developed during prototyping which further validates both the language and the model
- Language architecture is refined and clarified.
 - SysML is precisely specified as UML model (UML2 metamodel subset + profile + model library) which ensures all the virtues of MDA
 - Refactored package structure facilitates maintenance and language evolution
 - UML dependencies are precisely specified, which assists implementation and maintenance

Top-Level Changes (cont'd)

- Alignment with other standards and best practices is increased
 - IEEE-Std-1471-2000 (IEEE Recommended Practice for Architectural Description of Software-Intensive Systems)
 - IEEE Std. 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)
 - UML 2.0 Testing Profile Specification (OMG ptc/04-04-02).
- Information accessibility is significantly improved
 - Specification has been reorganized and rewritten to improve readability and consistency.
 - Indexes, figure lists, table lists, and cross-references have been added to improve navigation.

Detailed Changes (1/3)

- See *SysML v. 1.0a Reviewer Guide* for detailed changes
- Highlights
 - **Improvements to Structural Constructs**
 - Classes and Assemblies have been unified using the Block structural construct.
 - Flow Ports and Flow specifications have been added to specify input and output items.
 - may include data as well as physical entities, such as fluids, solids, gases, and energy.
 - Parametric Constraints are defined by extending UML Collaborations, which provide more natural semantics and distinctive notation for this new diagram type.
 - Added benefit: also supports pattern-based modeling

Detailed Changes: Highlights (2/3)



■ Improvements to Behavioral Constructs

- Activities have been refined to reduce their complexity and increase consistency with the rest of the specification.
- Interactions has been reduced to a subset of Sequence diagrams, which decreases the semantic overlap with Activities and simplifies the language.

■ Improvements to Cross-Cutting Constructs

- Requirement model element has been enhanced so that users can customize and assign classification categories, risks and verification methods.
- Allocation trace dependencies have been unified and simplified, and the content for tabular format is described in a non-prescriptive manner.
- Model management constructs have been added and include support for views and viewpoints in a manner compatible with IEEE-Std-1471-2000 (*IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*).
- Normative types and enumerations used by the SysML profile are modularly defined in a separate package and can be easily extended by the user without the need to create a new profile.

Detailed Changes: Highlights (3/3)



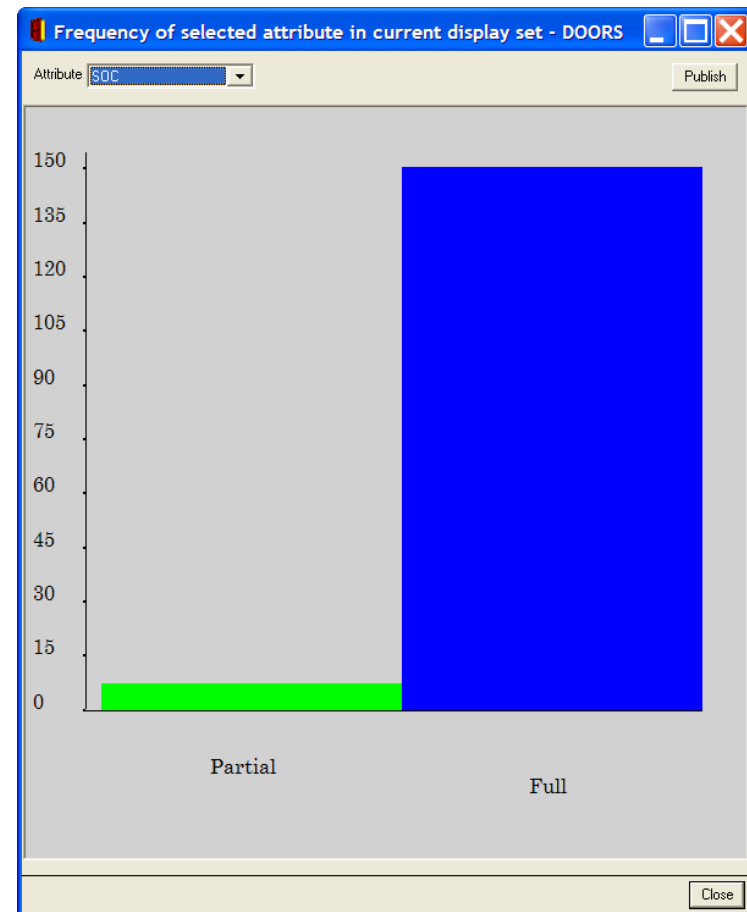
■ Other improvements

- The complete abstract syntax for the SysML profile and the UML 2.0 metamodel reused by the profile are provided to facilitate understanding, validate architecture integrity, and facilitate implementation and model interchange using XMI and AP-233.
- Non-normative extensions and model libraries are defined in separate appendices.
- The Requirements Traceability Matrix for tracking SysML compliance with the UML for SE RFP is now generated with a requirements management tool, so the information is more complete, accurate and consistent.

Statement of Compliance and Requirements Traceability

Statement of Compliance and Requirements Traceability

- Proposal satisfies all RFP mandatory requirements and most “optional” requirements
 - 157 mandatory requirements, section 6.5
 - Fully compliant with 150
 - Partially compliant with 7
 - 24 optional requirements, section 6.6
 - Fully compliant with 20
 - Partially compliant with 4



Requirements Traceability Matrix

OMG UML for Systems Engineering RFP	Req?	SOC	Requirement Satisfaction	Chapter Reference	Abstract and Concrete Syntax Reference	Sample Problem Reference
6.5.4.3 Requirement relationships						
UML for SE shall provide the capability to associate a requirement to one or more model elements, which include associations between:	True	Full	SysML defines a number of standard relationships between requirements and between requirements and other model elements. These are all forms of traceability (based on trace dependency). Standard relationships include: satisfy, verify, derive, allocate. Furthermore, requirements can be decomposed into sub-requirements.	Requirements	Table 18 Graphical Nodes for Requirements Table 19 Graphical Paths for Requirements	Figure B-30 Sample Traceability Matrix Figure B-29 Requirement Diagram: Requirement Traceability
a) Derived requirements and their source requirements (trace)	True	Full	SysML provides standard derive trace relationship.	Requirements	Table 18 Graphical Nodes for Requirements ...	Figure B-8 Requirement Diagram ...

Requirements Traceability Matrix

Formal module '/SysML/UML for Systems Engineering RFP' current 0.1 - DOORS

File Edit View Insert Link Analysis Table Tools User TAU Help

Mandatory Req SOC All levels

DMG UML for Systems Engineering RFP	Requirement	SOC	Requirement Satisfaction	Chapter Reference	Abstract and Concrete Syntax Referen	Sample Problem Reference
6.5.4.3 Requirement relationships						
UML for SE shall provide the capability to associate a requirement to one or more model elements, which include associations between:	True	Full	SysML defines a number of standard relationships between requirements and between requirements and other model elements. These are all forms of traceability (based on trace dependency). Standard relationships include: satisfy, verify, derive, allocate. Furthermore, requirements can be decomposed into sub-requirements.	Requirements	Table 17 Graphical Paths for Requirements Table 16 Graphical Nodes for Requirements	Figure B-29 Requirement Diagram: Requirement Traceability Figure B-30 Sample Traceability Matrix
a) Derived requirements and their source requirements (trace)	True	Full	SysML provides standard derive trace relationship.	Requirements	Table 17 Graphical Paths for Requirements Table 16 Graphical Nodes for Requirements	Figure B-8 Requirement Diagram: Requirements Derivation
b) Requirements and the model elements that realize and/or implement the requirements	True	Full	SysML provides standard allocate trace relationship and satisfy trace relationship.	Requirements	Table 17 Graphical Paths for Requirements Table 16 Graphical Nodes for Requirements	Figure B-28 Requirement Diagram: Requirement Satisfaction
c) Requirements and goals of a system by hierarchical decomposition into lower level requirements and sub-goals	True	Full	SysML support the hierarchical decomposition of requirements via composition association.	Requirements	Table 17 Graphical Paths for Requirements Table 16 Graphical Nodes for Requirements	Figure B-1 Requirement Diagram: Top-Level User Requirements

Username: Chris Sibbald Exclusive edit mode

Verification & Validation

Verification & Validation



- Proof of concept (RFP section 4.8)
 - TAU G2 v. 2.6 GA implements SysML v. 1.0a specification
(see demo at this meeting)
 - other SysML Partner vendor implementations TBA
- Other verification & validation
 - abstract syntax precisely specified using UML v. 2.0 compliant modeling tool
 - INCOSE MDSD WG, ADTF review processes
 - public feedback from SysML web and SysML Forum mailing list

Proof of Concept: Executable Sample Problem

Hybrid SUV

- The sample problem describes the use of SysML as it applies to the development of a Hybrid SUV
- Problem derived from marketing analysis which indicates a need to increase fuel economy and eco-friendliness of the vehicle, without sacrificing performance.

[Screenshots are from executable model developed to validate SysML v. 1.0a specification]

Requirements Diagram

- The Requirements diagram provides the following features:
 - requirement stereotype that can represent text based requirements and properties (e.g., id, text statement)
 - requirements can be decomposed into their constituent elements
 - requirements can be sub-classed using specialization
 - derive relationship between derived and source requirements
 - satisfy relationship between design elements and requirements
 - verify relationship between requirements and test cases
 - rationale for requirements traceability, satisfaction, etc.

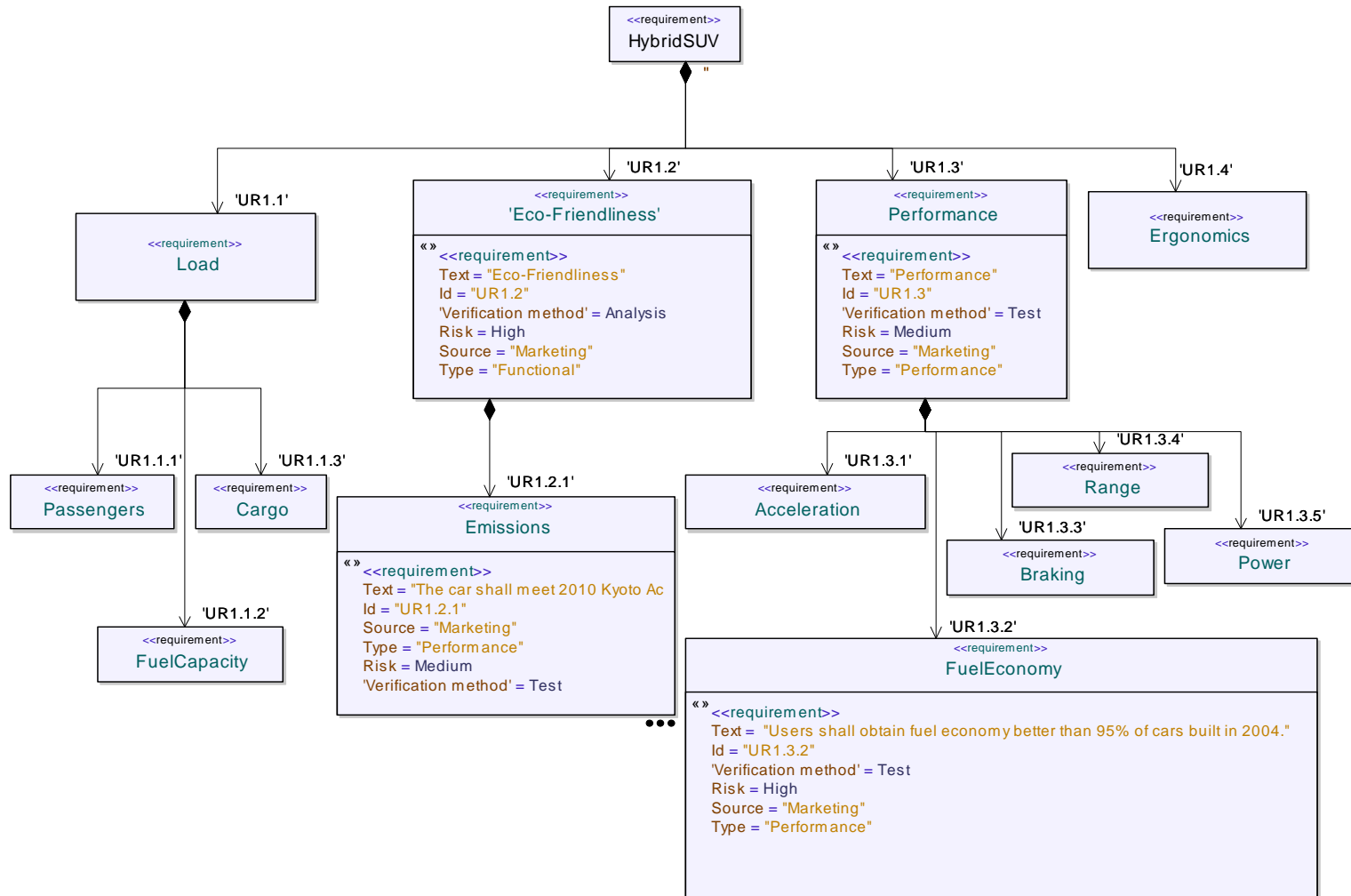
Top Level User Requirements



Top-Level Requirements

package Requirements {1/5}

IG LANGUAGE



Requirement Report



MODELING LANGUAGE

SysML-Auto-Example.ttw - Telelogic TAU - [Top-Level Requirements]

File Edit View Project Verify Build Link Tools Window SysML Help

- Show SysML Dependency Matrix
- Show SysML Dependency Report
- Save SysML Dependency Report File
- Show Requirements Report
- Show Requirements Gap Report
- Update Allocated

Risk = Medium
Source = "Marketing"
Type = "Performance"

UR1.3.1 Acceleration

UR1.3.2 FuelEconomy

UR1.3.3 Braking

UR1.3.4 Range

UR1.3.5 Power

Element: FuelEconomy Options...

Filter: Requirement Stereotypes...

Text: Users shall obtain fuel economy better than 95% of cars built in 2004.

Id: UR1.3.2

Verification method: Test

Risk: High

Source: Marketing

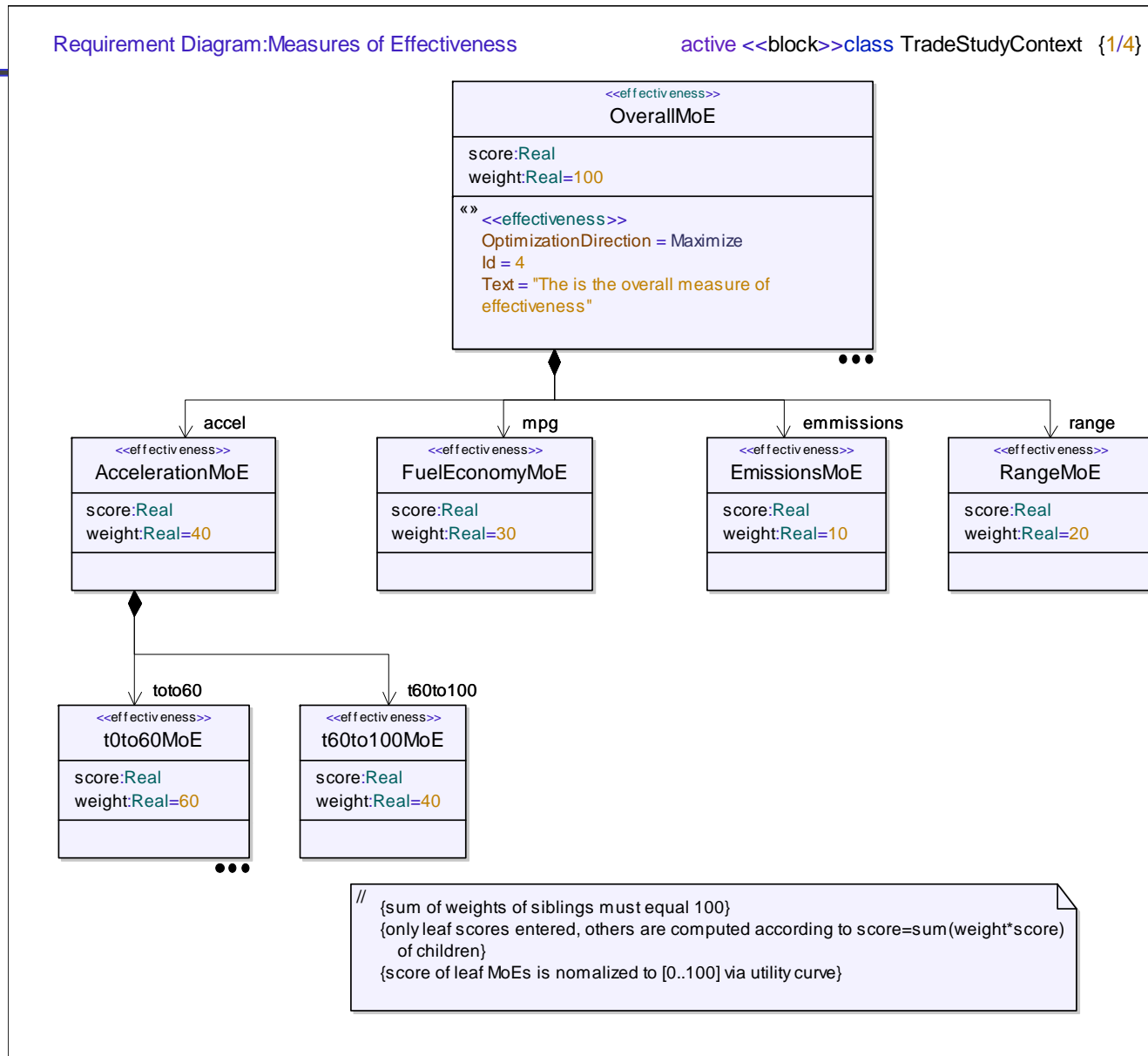
Type: Performance

Id	Text	Verification method	Risk	Source	Type
Load	Load	Test	Low	Marketing	Functional
Eco-Friendliness	Eco-Friendliness	Analysis	High	Marketing	Functional
Emissions	The car shall meet 2010 Kyoto Accord emissions standards.	Test	Medium	Marketing	Performance
Performance	Performance	Test	Medium	Marketing	Performance
FuelEconomy	Users shall obtain fuel economy better than 95% of cars built in 2004.	Test	High	Marketing	Performance
Regenerative...	The vehicle shall convert kinetic energy to electrical energy to recharge the battery and provide ad...	Test	Medium	Trade Study 13-25	Function
PowerSource...	If the vehicle velocity is below 10 km/h the electric motor shall be used as the sole power source. □...	Test	Medium	Trade Study 13-25	Functional

SysML: Generate and show a SysML requirements report in the Output window

start | C:\Laptop Backup-1\... | SysML-Auto-Example... | Microsoft PowerPoint... | 9:39 AM

Trade Studies and MoE



Key Performance Parameters

Internal Block Diagram:Key Performance Parameters active <<block>>class TradeStudyContext {2/4}

::SUV::Blocks::HybridSUV::mass

::SUV::Blocks::ElectricalMotor::maxHP

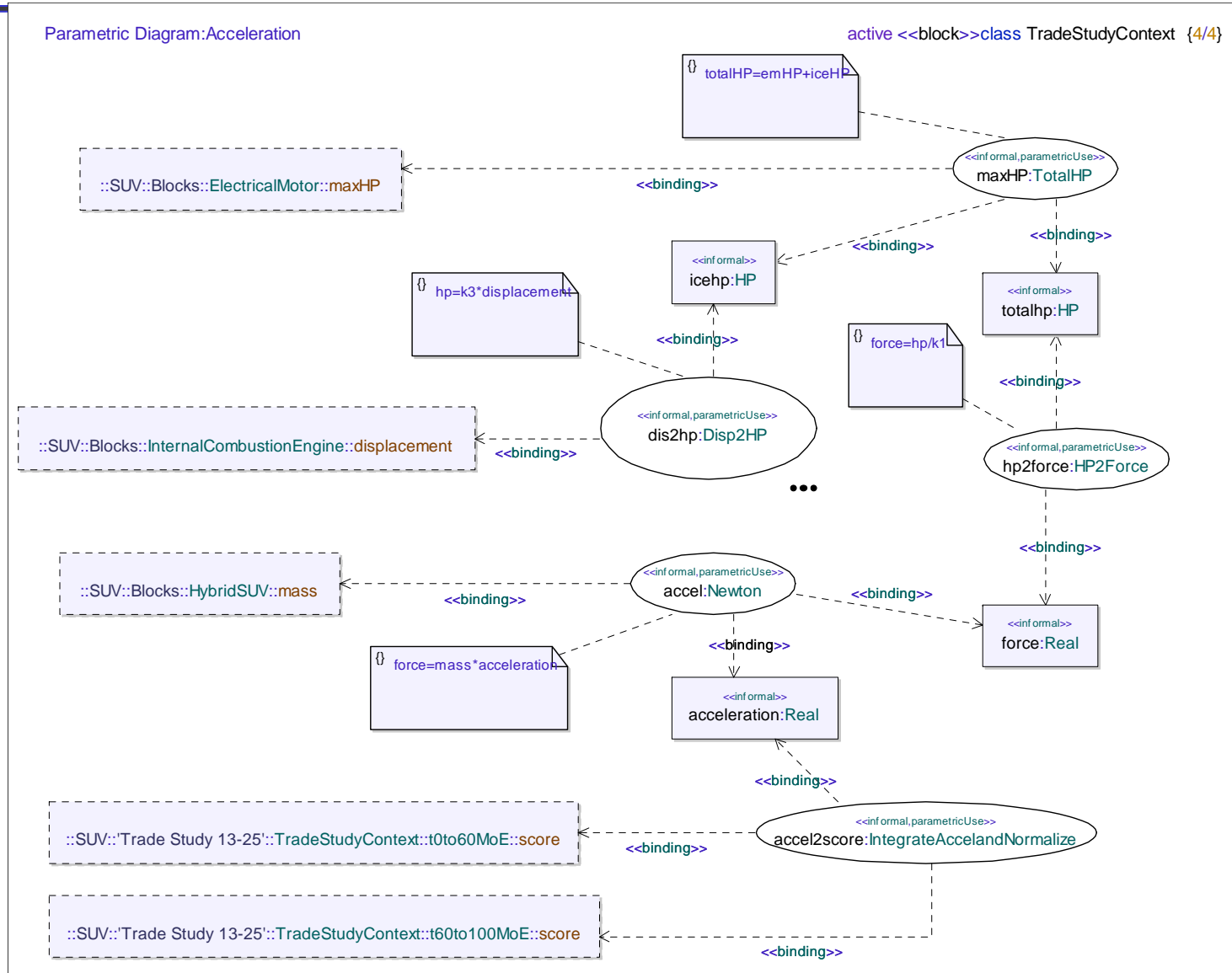
::SUV::Blocks::BatteryPack::energyDensity

::SUV::Blocks::InternalCombustionEngine::displacement

::SUV::Blocks::FuelTank::capacity

::SUV::Blocks::ChassisSubsystem::dragcoef

Parametric Constraints for Mapping KPP to Acceleration MoE



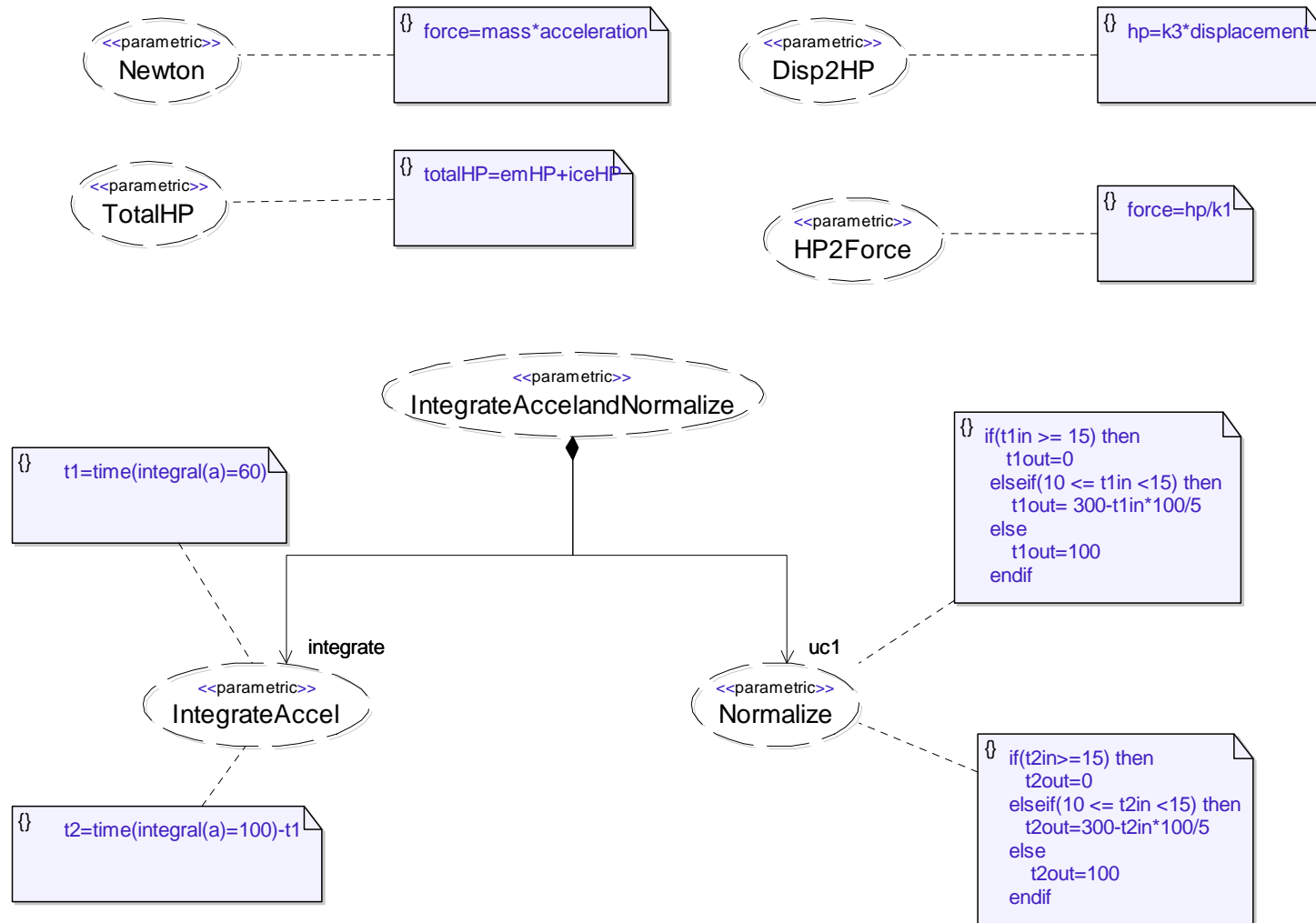
Parametric Constraint Definitions



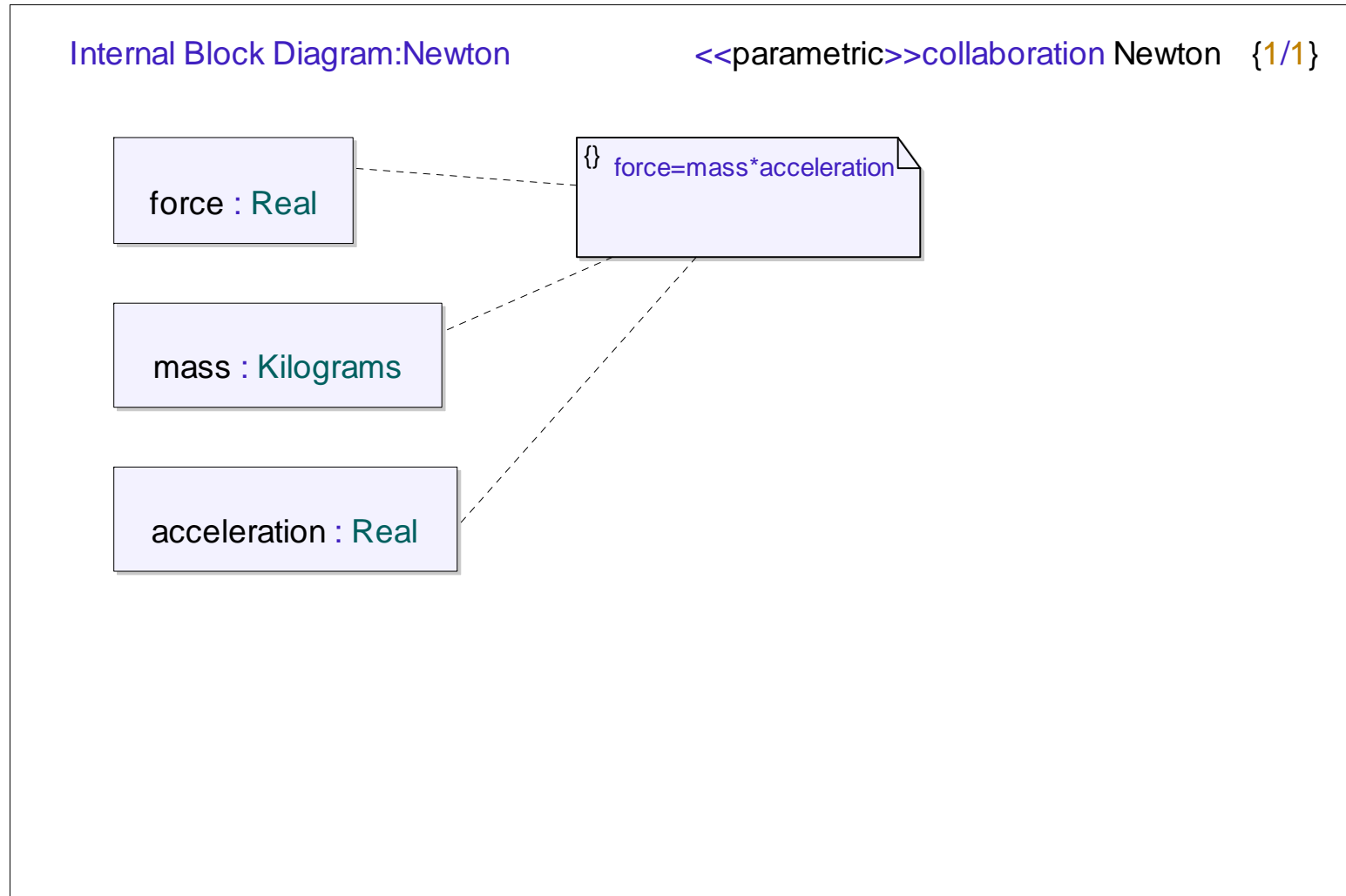
SYSTEMS MODELING LANGUAGE

Block Definition Diagram:Constraint definitions

active <<block>>class TradeStudyContext {3/4}



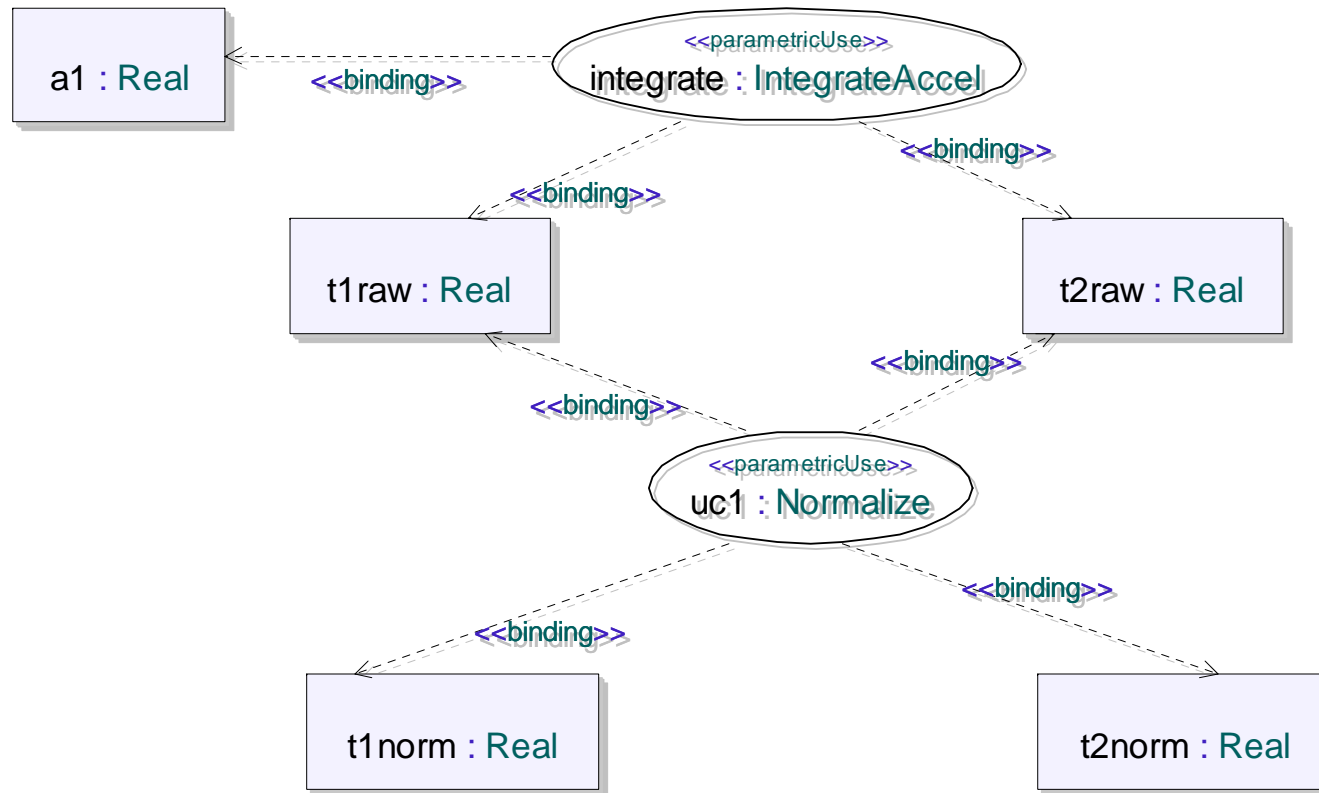
Parametric Constraint Definition: Newton's 2nd Law



Parametric Constraint Definition: Composite Constraint

Internal block diagram1

`<<parametric>>collaboration {1/1}`
`IntegrateAccelandNormalize`

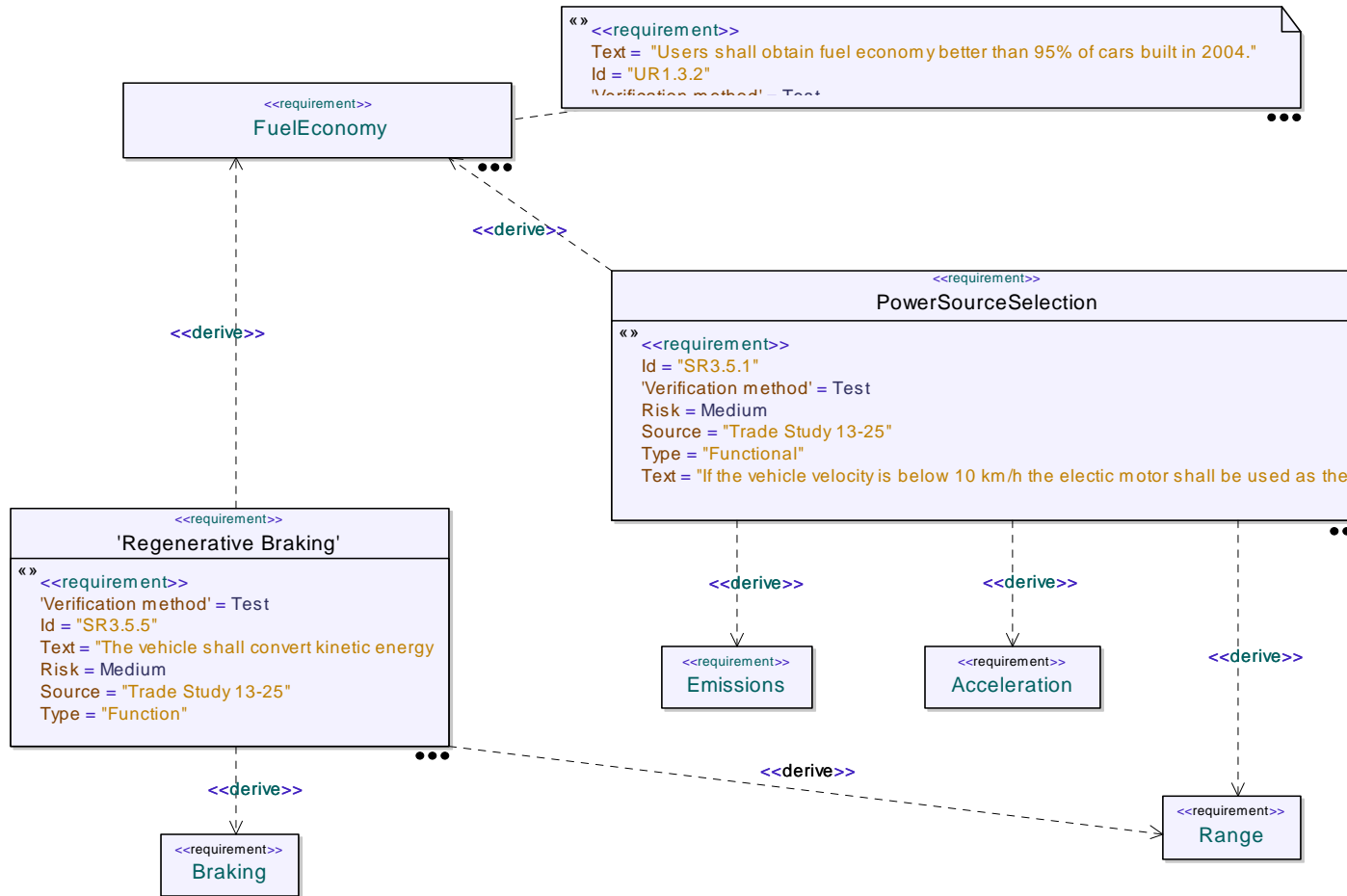


Requirements Derivation



Requirement Derivations

package Requirements {2/5}

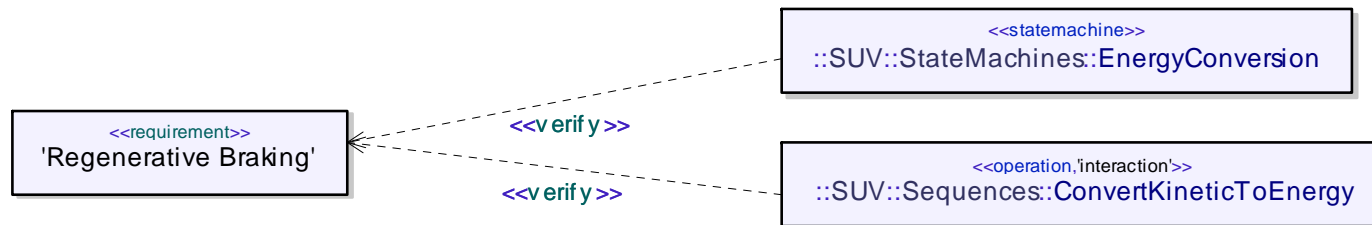


Requirements Verification

Requirement Verification

package Requirements {4/5}

```
// NOTES  
1) Requirements may be verified by behaviors(sequence diagrams, activity diagrams and statemachines  
and other procedural specifications for analysis, inspection, demonstration methods of verification.
```

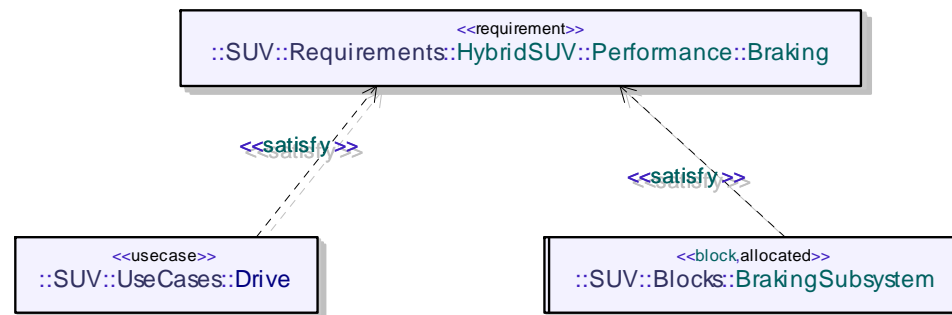


Requirements Satisfaction

Requirement Satisfaction

package Requirements {3/5}

// NOTES
1) Use Cases, like any other structure or behavior,
can <<satisfy>> Requirements



Use Case Diagram

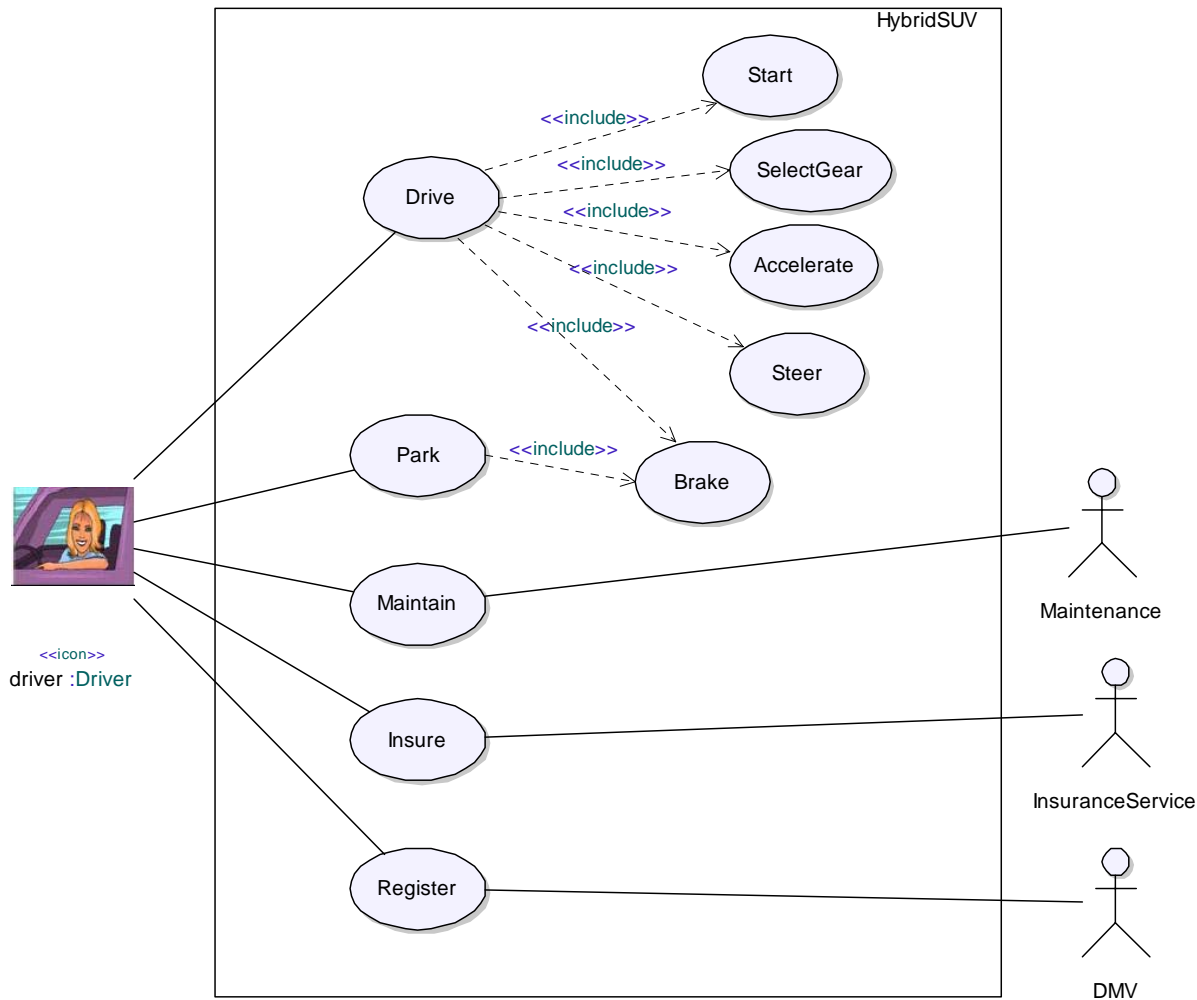
- System uses
 - represents mission capabilities and/or uses of a system that are implemented by system
 - shows external context for system (other systems, operators, etc.)
 - simplifies understanding of context and external interfaces

Use Cases – Context Level



Top-Level Use Cases

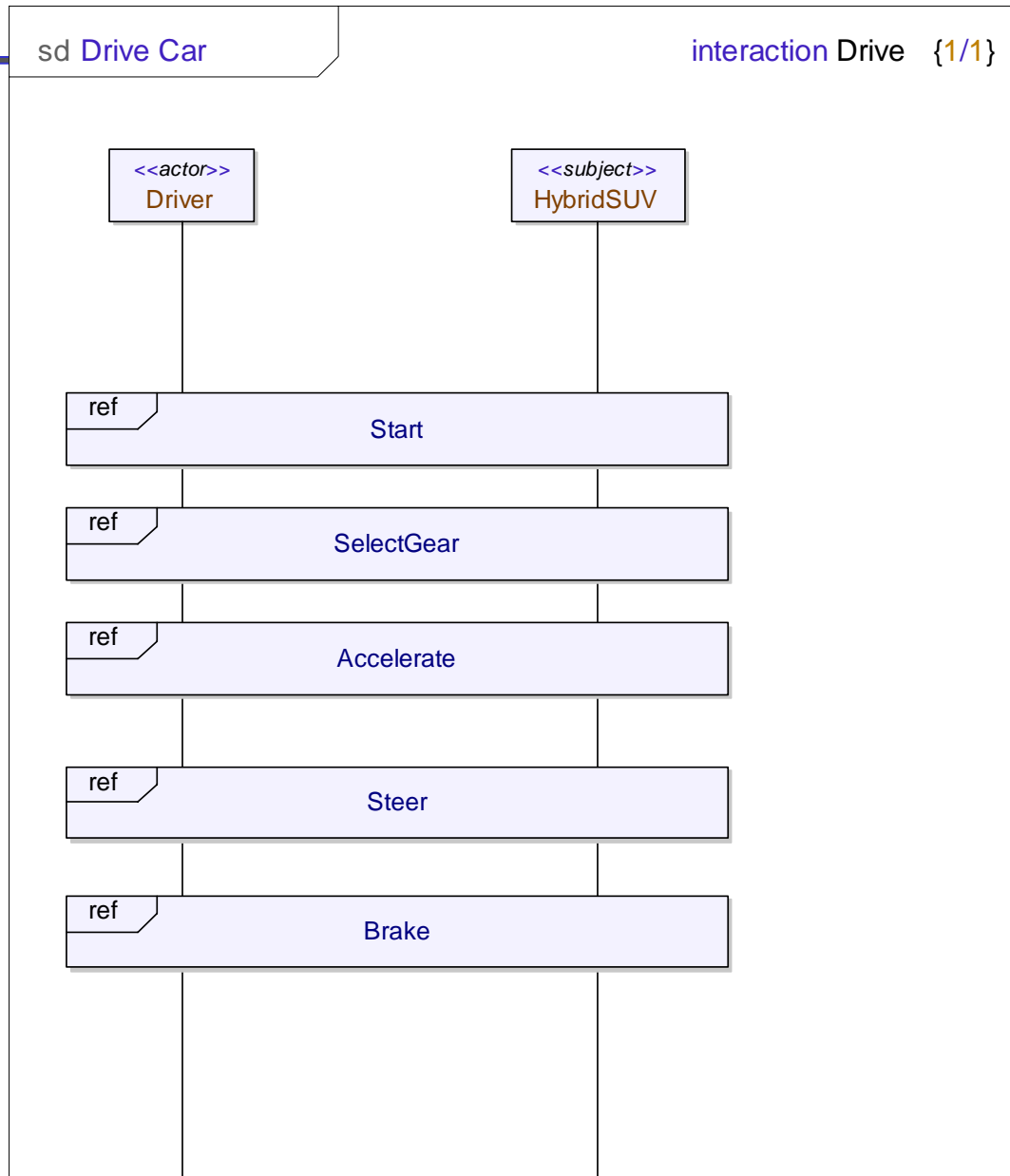
package UseCases {1/1}
SYSTEMS MODELING LANGUAGE



Sequence Diagram

- Sequence diagrams show the parts of the system via lifelines (or a single lifeline representing the subject system) and messages that are exchanged between the environment (actors) and the lifelines.
- Shows control and data flow
- Useful for analyzing key system scenarios and response threads.
- One can show states and activities on the lifeline to show which functions are invoked and/or which state transitions occur.
- Interaction Fragments can be used to “decompose” and abstract scenarios.
- Sequence diagrams can be used to specify behavior that is required or automatically generated by executing the model in order to validate behavior.

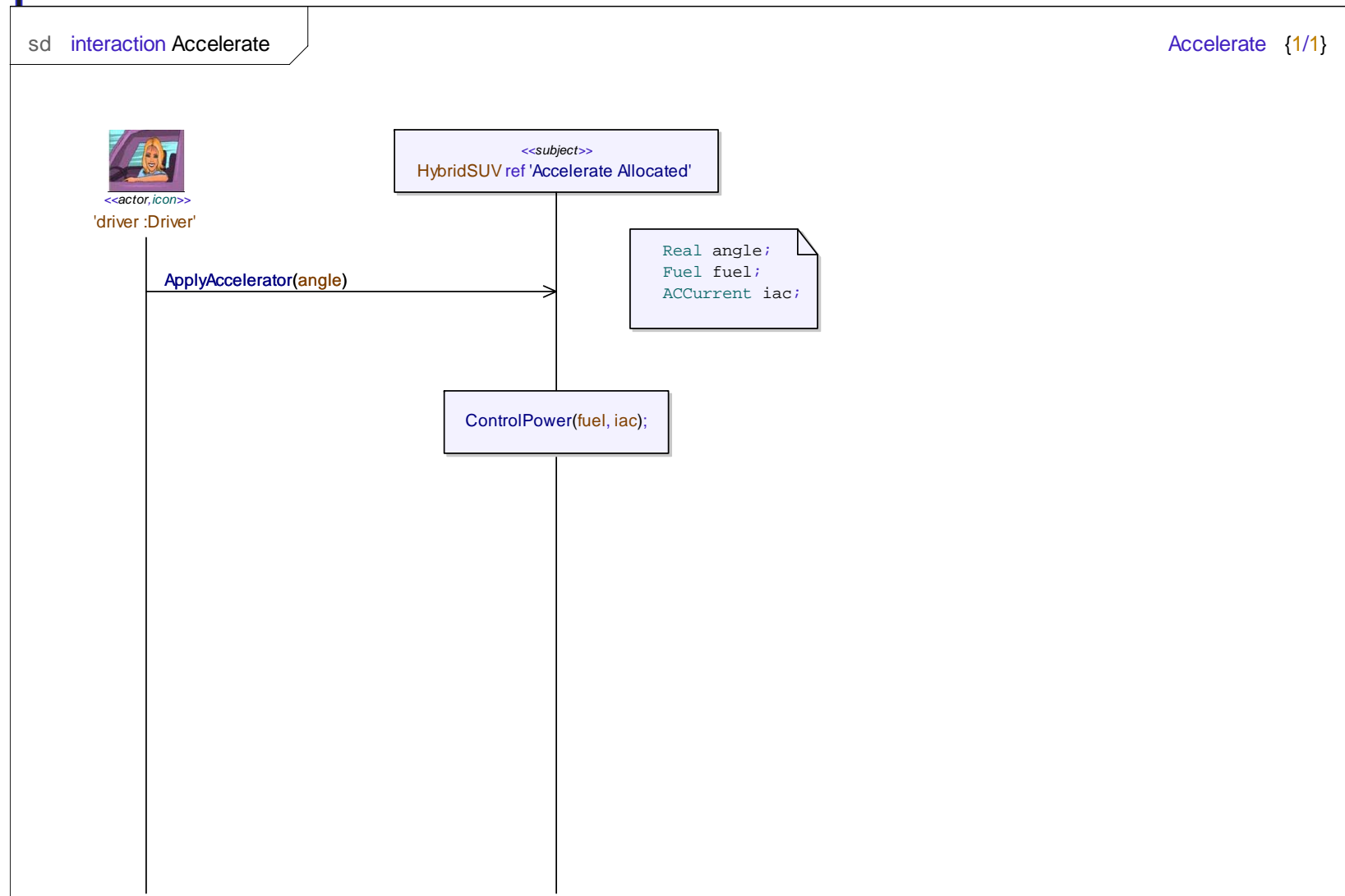
Scenario – Context Level



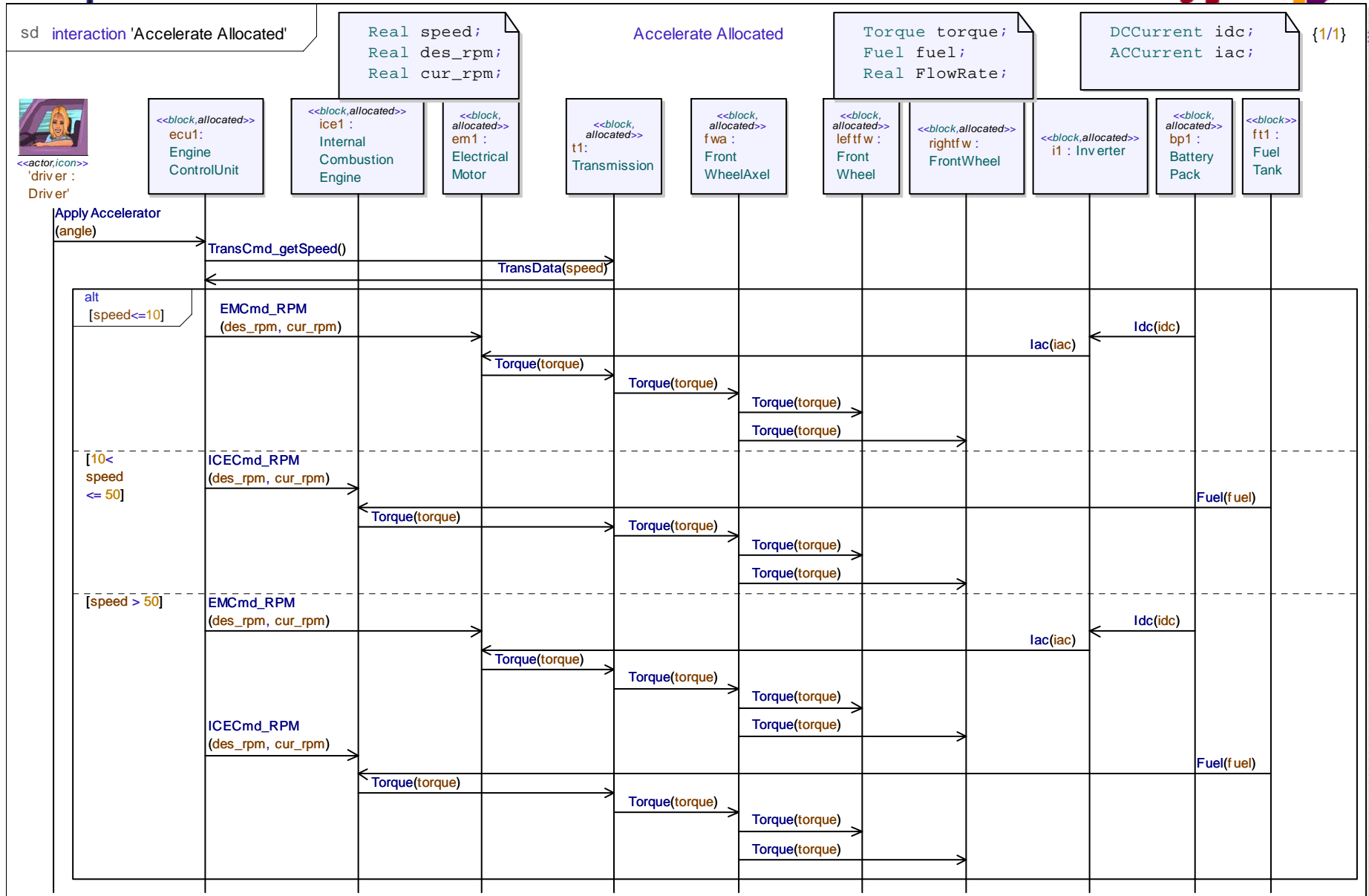
"Black-box" Scenario – Accelerate



SYSTEMS MODELING LANGUAGE



Allocated Scenario – Accelerate (alternate behavior)



Activity Diagram

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

Activity – Control Power

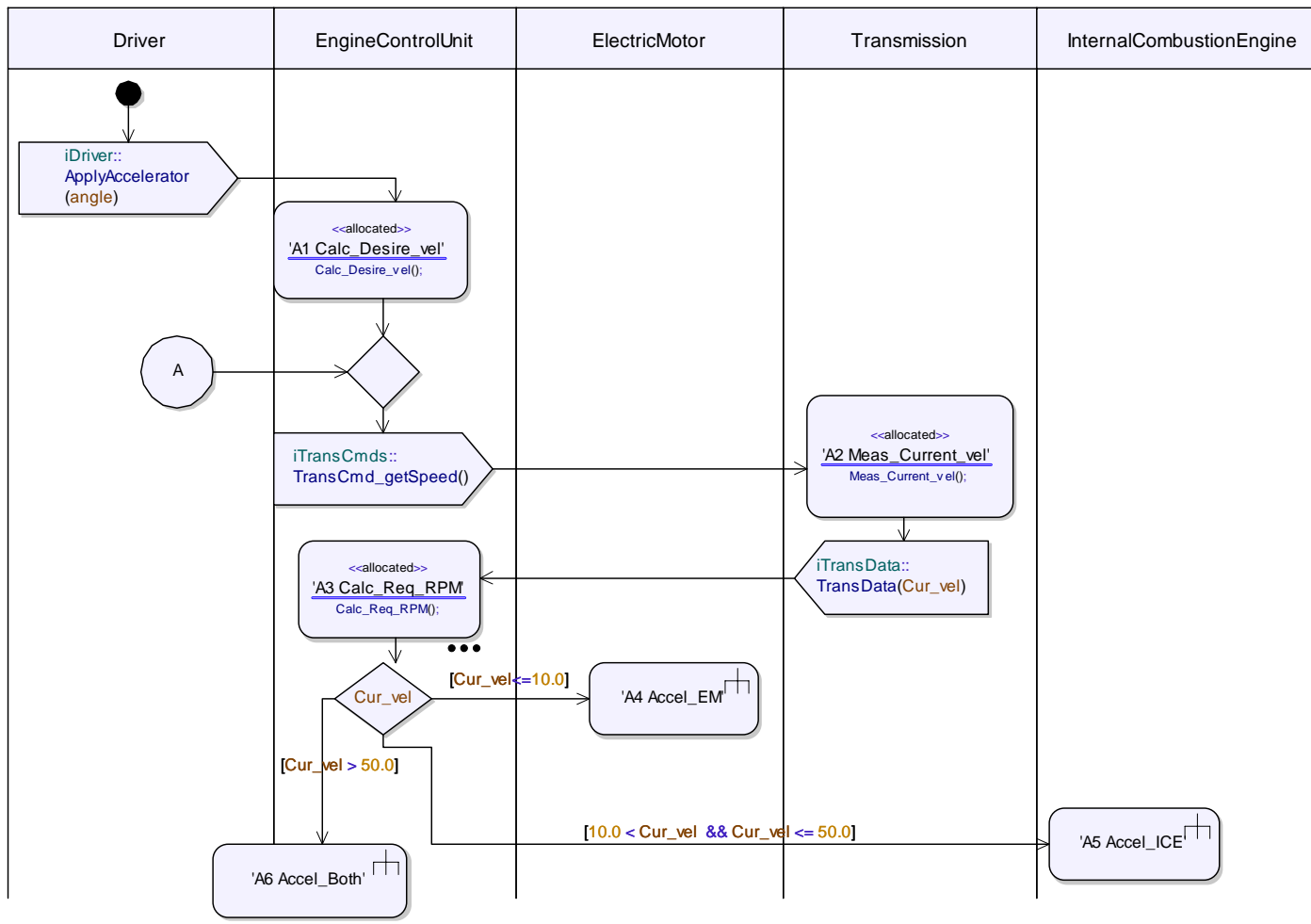


activity ControlPower {1/1}

act: Control Power

```
Real angle;
Real Des_vel;
Real Cur_vel;
Real des_rpm;
Real cur_rpm;
Torque t;
```

```
Real flowrate
Fuel fuel;
DCCurrent Idc;
```



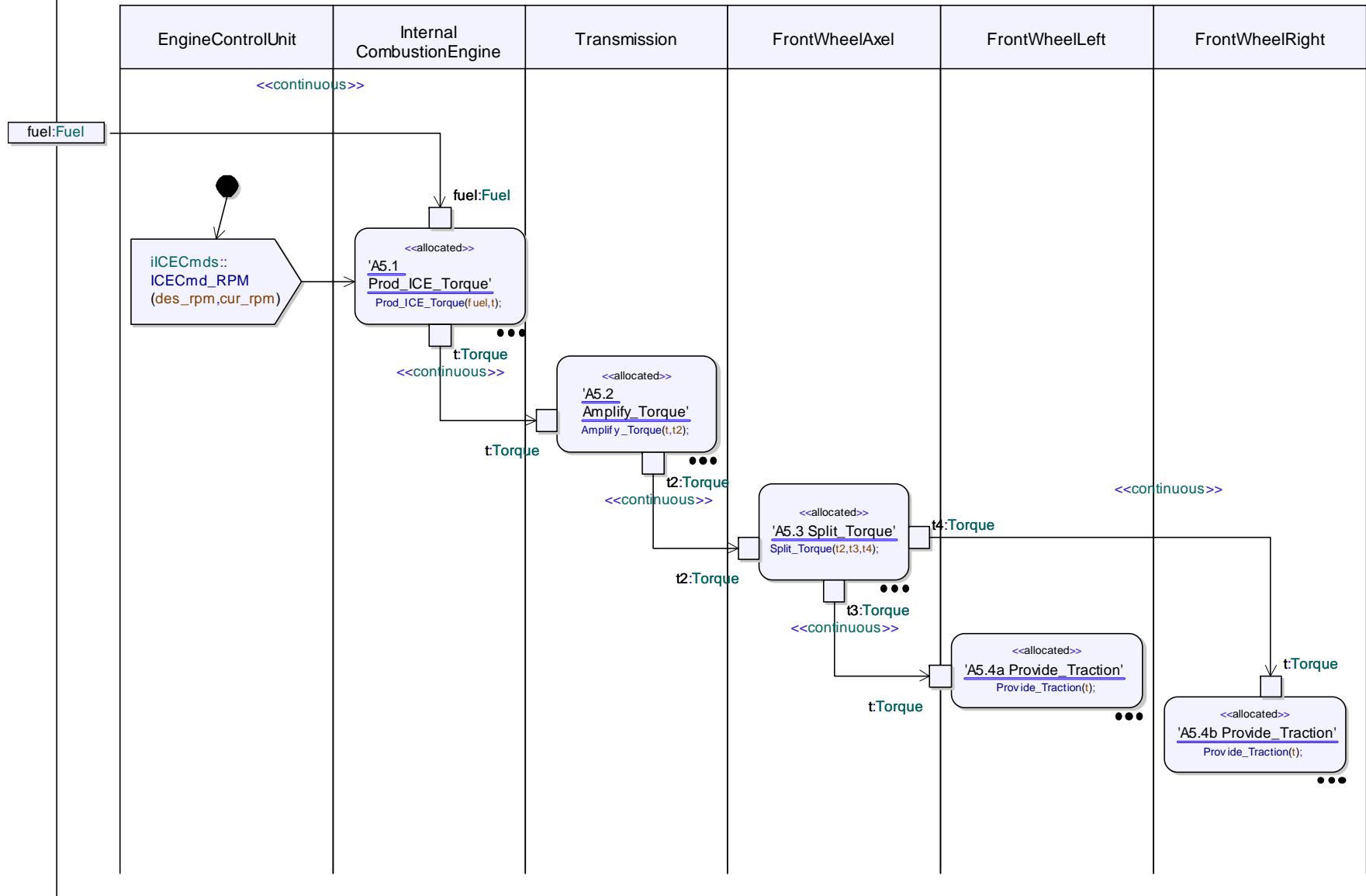
Activity – Control Power ($10 < v \leq 50$)



Activity Diagram:Accelerate ICE

activity 'A5 Accel_ICE' {1/1}

JAGE

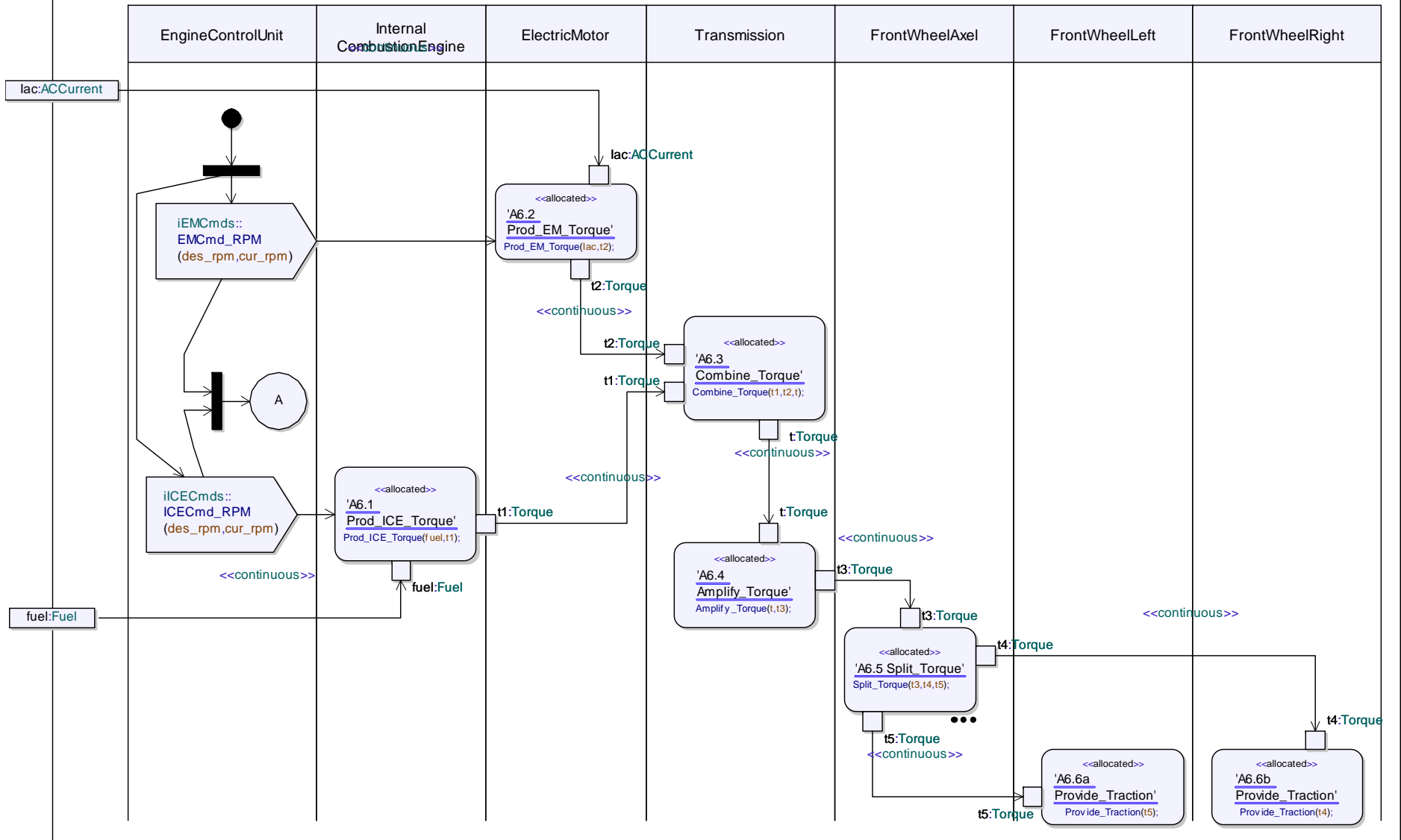


Activity – Control Power ($v > 50$)



Activity Diagram: Accelerate Both Power Sources

activity 'A6 Accel_Both' {1/1}



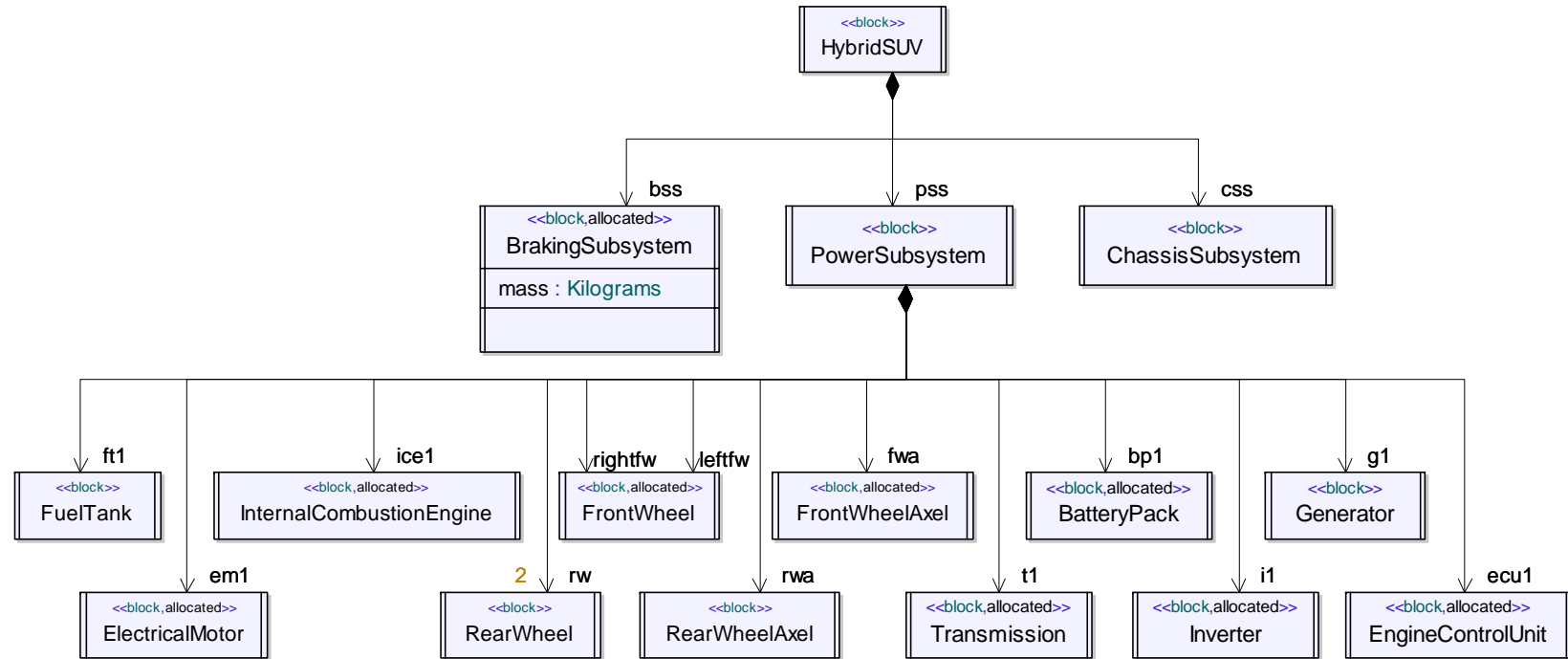
Block Diagrams

- Structural modeling foundation
- Blocks are UML 2 structured classes
 - Classes which support the ability to hold parts, ports, and internal connectors
- Blocks provide both black-box view (without internal structure) and white-box view (showing internal parts and connectors)
- Basis for allocation and refinement across multiple structure breakdowns (e.g., mechanical vs. electrical)
- Provide detail as needed to serve purpose at each development stage
 - initial sketch for communication
 - comprehensive detail for simulation/execution
- Compatible with features specific to software components (required/provided interfaces, ports with signaling behavior, etc.)

Block Definition Diagram - EBS

Equipment Breakdown Structure

package Blocks {1/2}



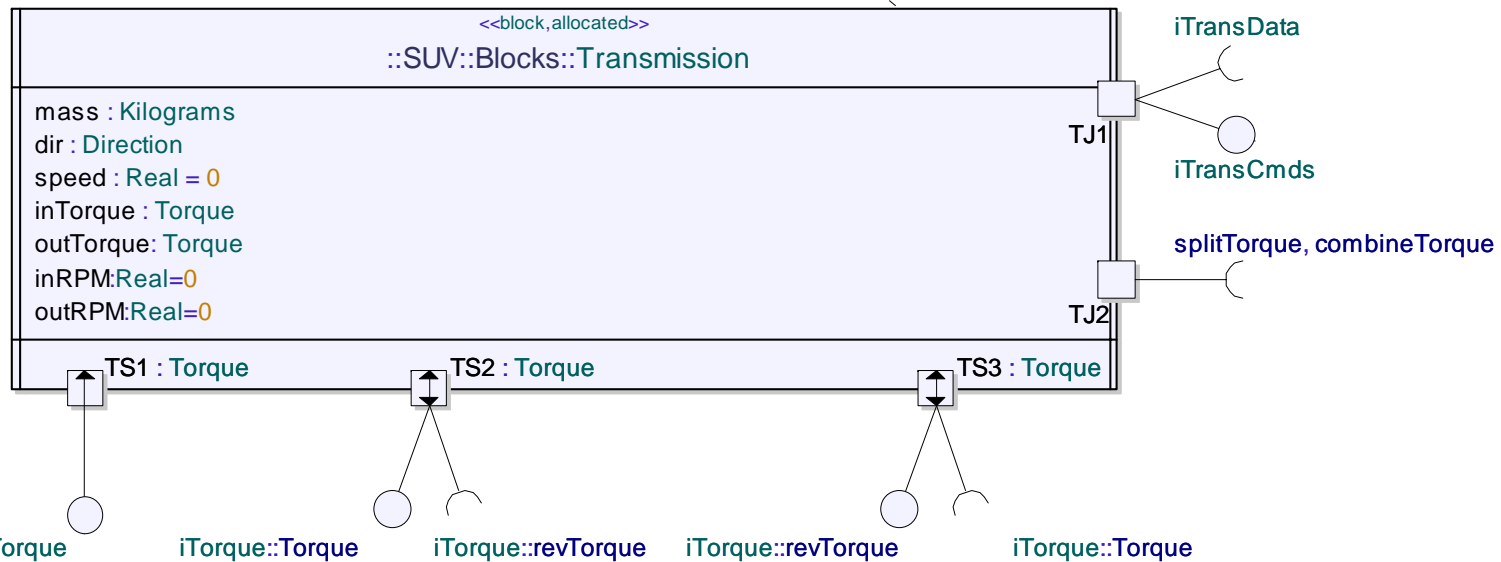
Properties of Transmission

Transmission Properties

active <<block,allocated>>class Transmission {2/2}

```

«» <<allocated>>
allocatedFrom = "Shift, A2 Meas_Current_vel, ConfigRegen, Amplify_RevTorque, A4.2 Amplify_Torque, A5.2 Amplify_Torque, A6.3
Combine_Torque, A6.4 Amplify_Torque"
    
```



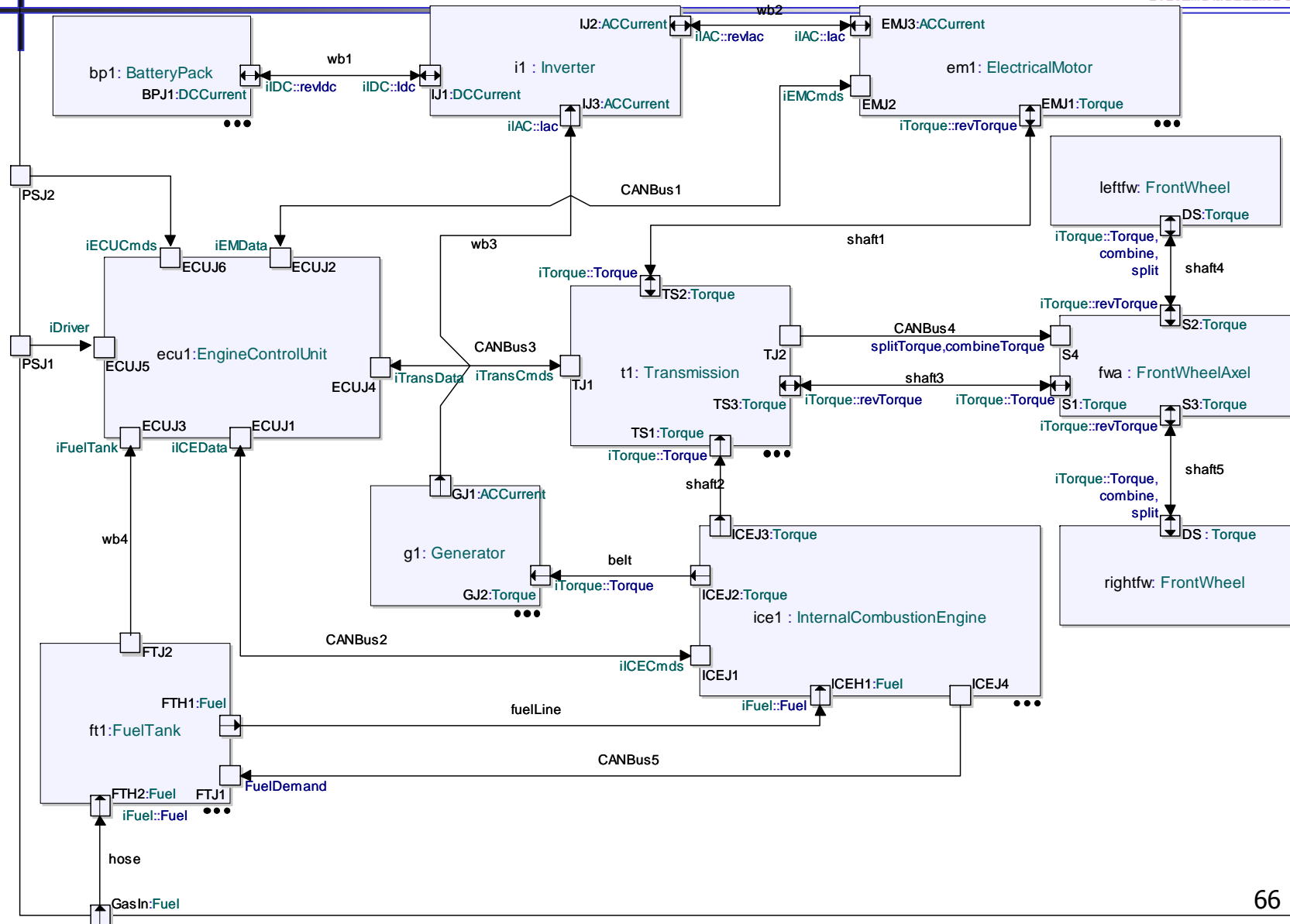
Internal Structure of Power Subsystem



ibd: Hybrid SUV Power Subsystem

active <<block>>class PowerSubsystem {1/1}

SYSTEMS MODELING LANGUAGE



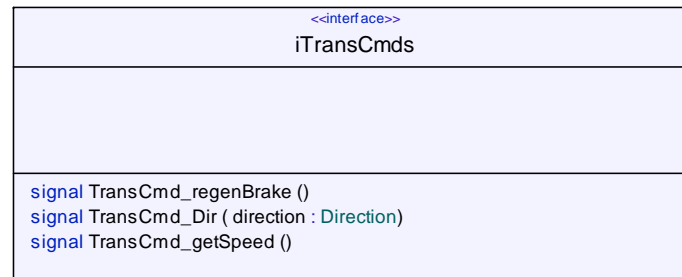
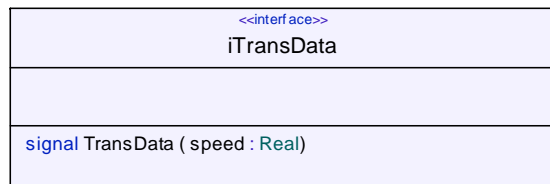
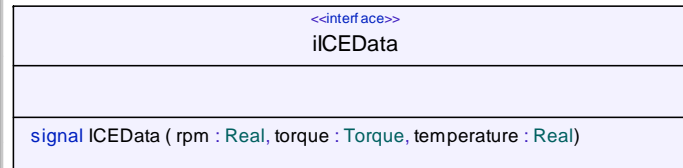
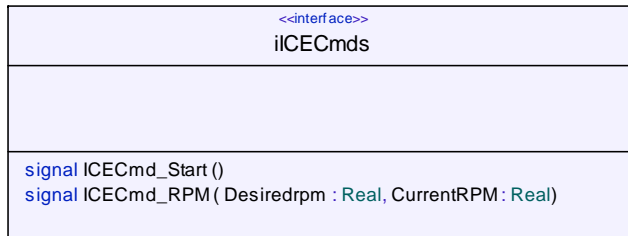
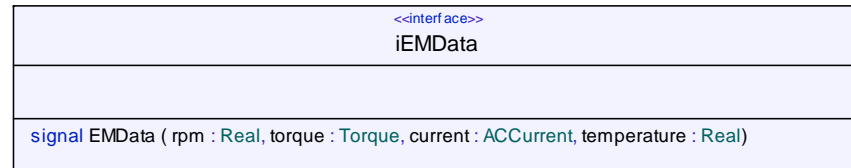
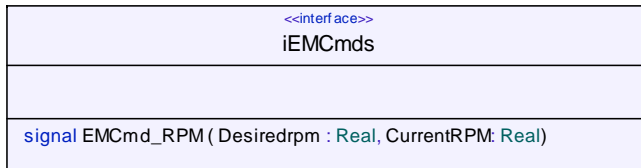
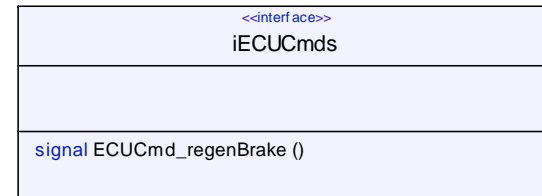
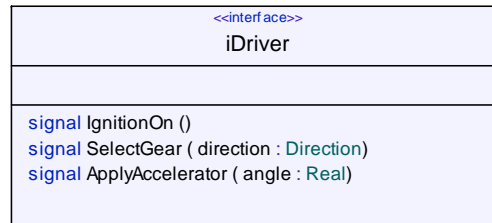
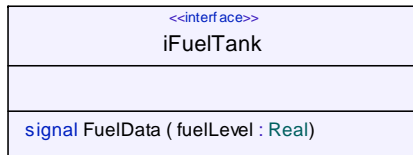
Command and Telemetry Interfaces Specifications



Command and Telemetry Interface Definitions

package Interfaces {1/8}

3 LANGUAGE

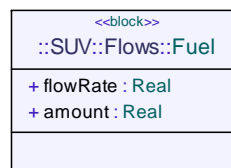
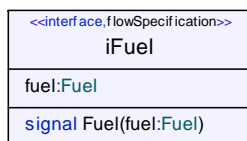
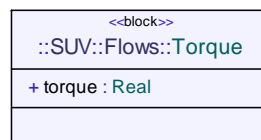
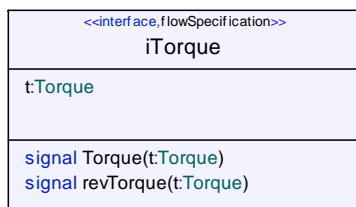
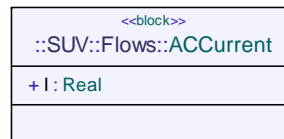
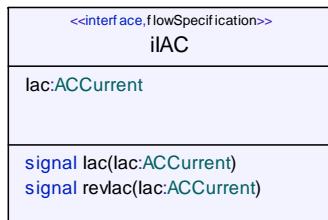
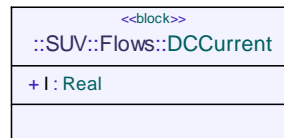
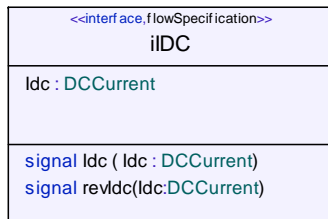


Flow Specifications



Flow Specification Definitions

package Interfaces {8/8}



Allocations

- Allocation is a key systems engineering concept that enables one set of model elements to be mapped to another set during design
- SysML provides support for multiple types of allocation

Uses of Allocation

Usage	From	To
1. Requirement Allocation	Requirement	Assembly/Part
2. Functional Allocation	Function (activity), or State	Assembly/Part
3. Logical Allocation	Assembly/Part, I/O (logical)	Assembly/Part, I/O (physical)
4. SW to HW	Assembly/Part (software)	Assembly/Part (hardware)
...

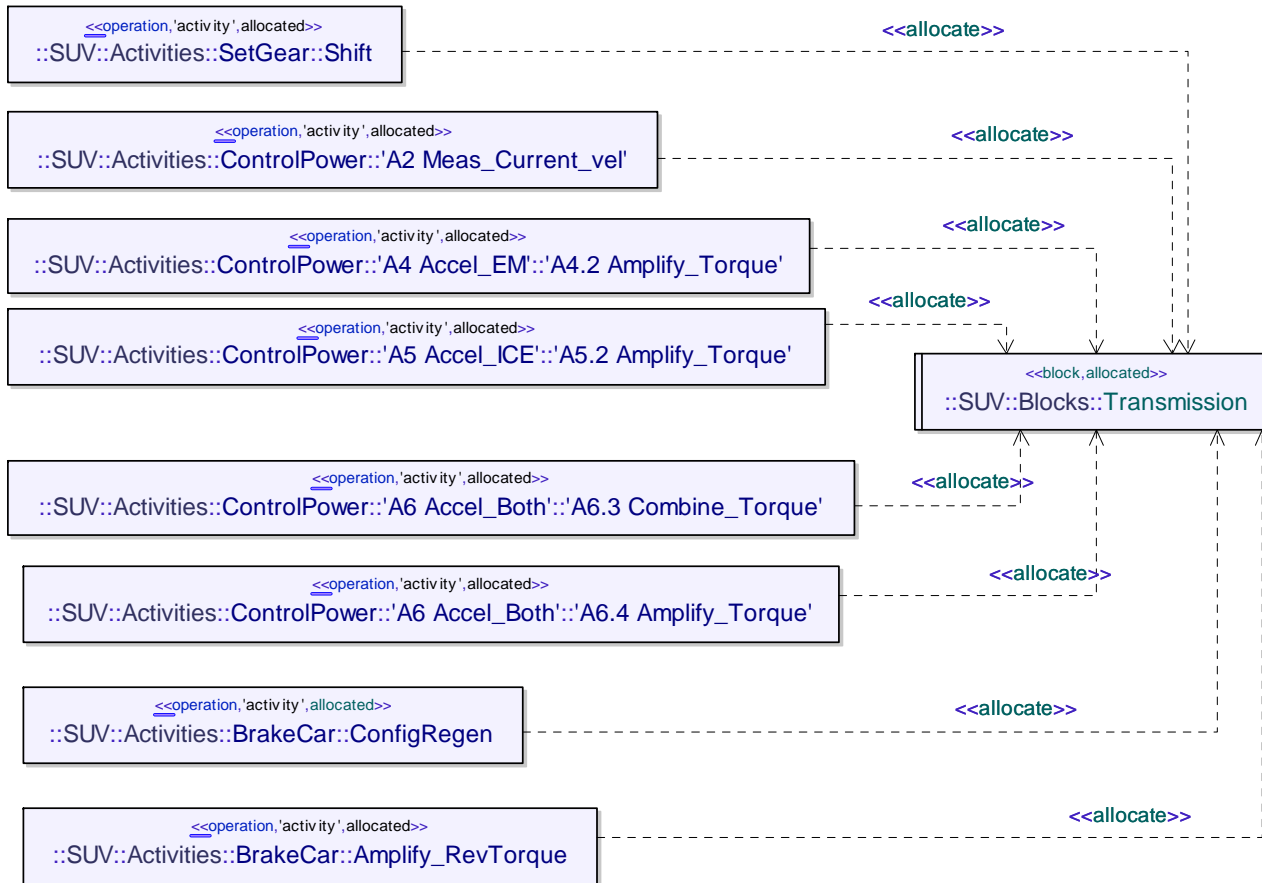
Functional Allocation to Transmission



bd: Func Allocations to Transmission

active <<block,allocated>>class Transmission {1/2}

LANGUAGE



Allocation Table



SYSTEMS MODELING LANGUAGE

SysML-Auto-Example.ttw - Telelogic TAU - [bd: Func Allocations to Transmission]

File Edit View Project Verify Build Link Tools Window SysML Help

Show SysML Dependency Matrix
 Show SysML Dependency Report
 Save SysML Dependency Report File
 Show Requirements Report
 Show Requirements Gap Report
 Update Allocated

All
 All - Reversed
 Allocate
 Allocate - Reversed
 Derive
 Derive - Reversed
 Satisfy
 Satisfy - Reversed
 Verify
 Verify - Reversed

active <<block,allocated>>class Transmission {1/2}

bd: Context

HybridSUV

BrakingSubsystem

ChassisSubsystem

InternalCombustionEngi

FrontWheel

Transmission

Dependencies

of Shift

of A2 Meas_Curr

of ConfigRegen

bd: Func Allocations

Transmission Proper

mass : Kilograms

dir : Direction

speed : Real

inTorque : Torque

inRPM : Real

outTorque : Torque

outRPM : Real

initialize()

File View Model View

Source EngineContro... InternalCom... Transmission ElectricalMotor FrontWheelA... FrontWheel BrakingSubsy... Inverter BatteryPack

SwitchGear	allocate								
Shift			allocate	allocate					
Prod_EM_Torque				allocate					
A1 Calc_Desire_vel	allocate								
A2 Meas_Current_vel			allocate						
A3 Calc_Req_RPM	allocate								
A4.1 Prod_EM_Torque				allocate					
A4.2 Amplify_Torque			allocate						
A4.3 Split_Torque					allocate				
A4.4a Provide_Traction						allocate		allocate	
A4.4b Provide_Traction								allocate	
A5.1 Prod_ICE_Torque		allocate							
A5.2 Amplify_Torque			allocate						
A5.3 Split_Torque					allocate				
A5.4a Provide_Traction							allocate		

SysML: Generate and show a SysML allocation matrix in the output window

start D:\Laptop Back... SysML-Auto-Ex... Microsoft Power... Adobe FrameMa... 3:59 PM

State Machine Diagram

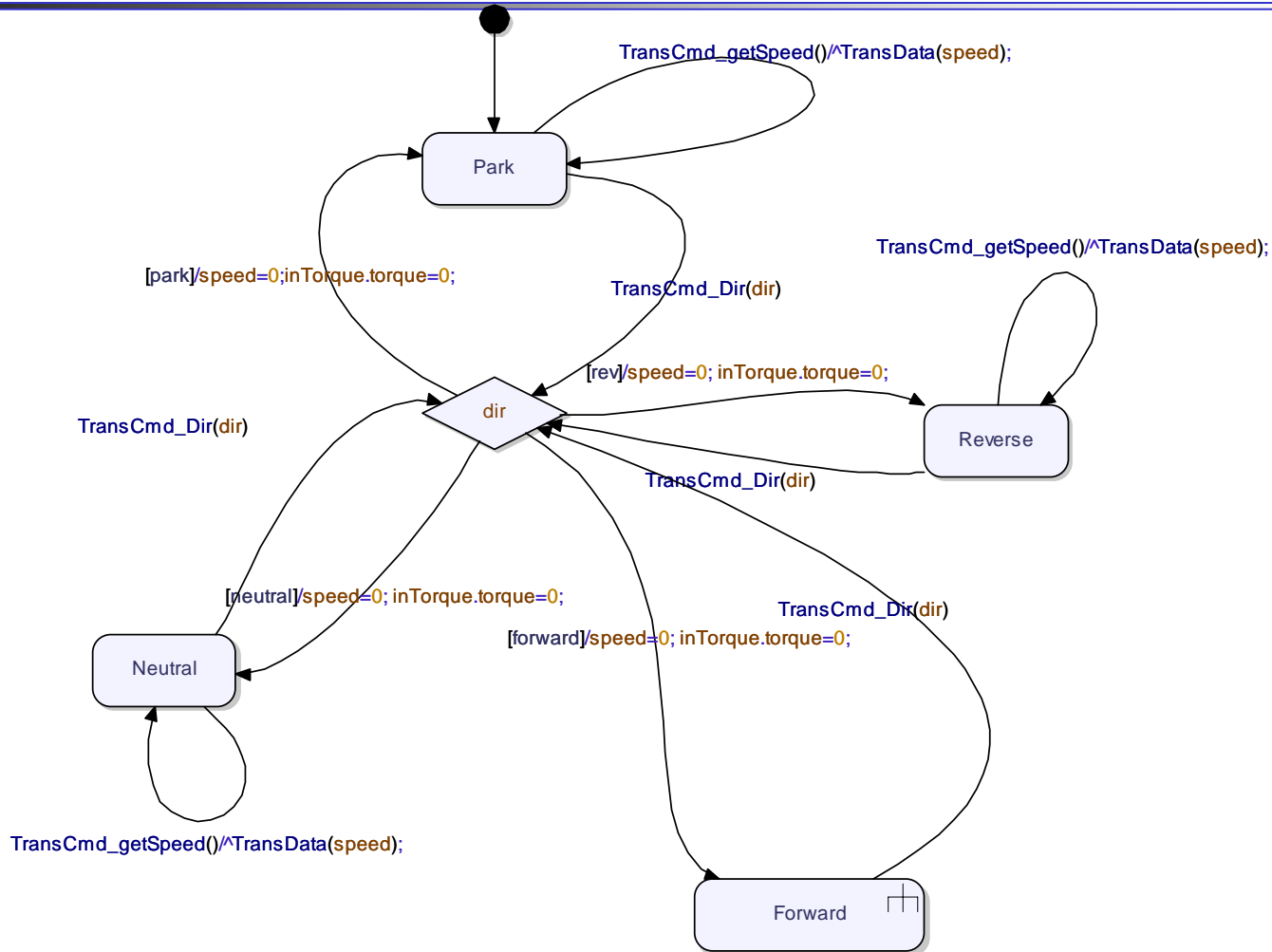
- State machine diagram capture
 - States of objects
 - Transitions between states:
 - Events causing transitions
 - Conditions under which a transition is taken (guard)
 - Actions (if any) taken when the transition happens
- Composite states are hierarchical
 - can contain 1 region or 2 or more orthogonal regions
- Objects can send/receive signals to communicate with other objects during state transitions, etc
- Used to implement dynamic behavior based on activity allocation
- Permits model execution to assess proper emergent behavior

State Machine Diagram – Shift



Shift

statemachine Transmission :: initialize(1)



Transmission State - Forward

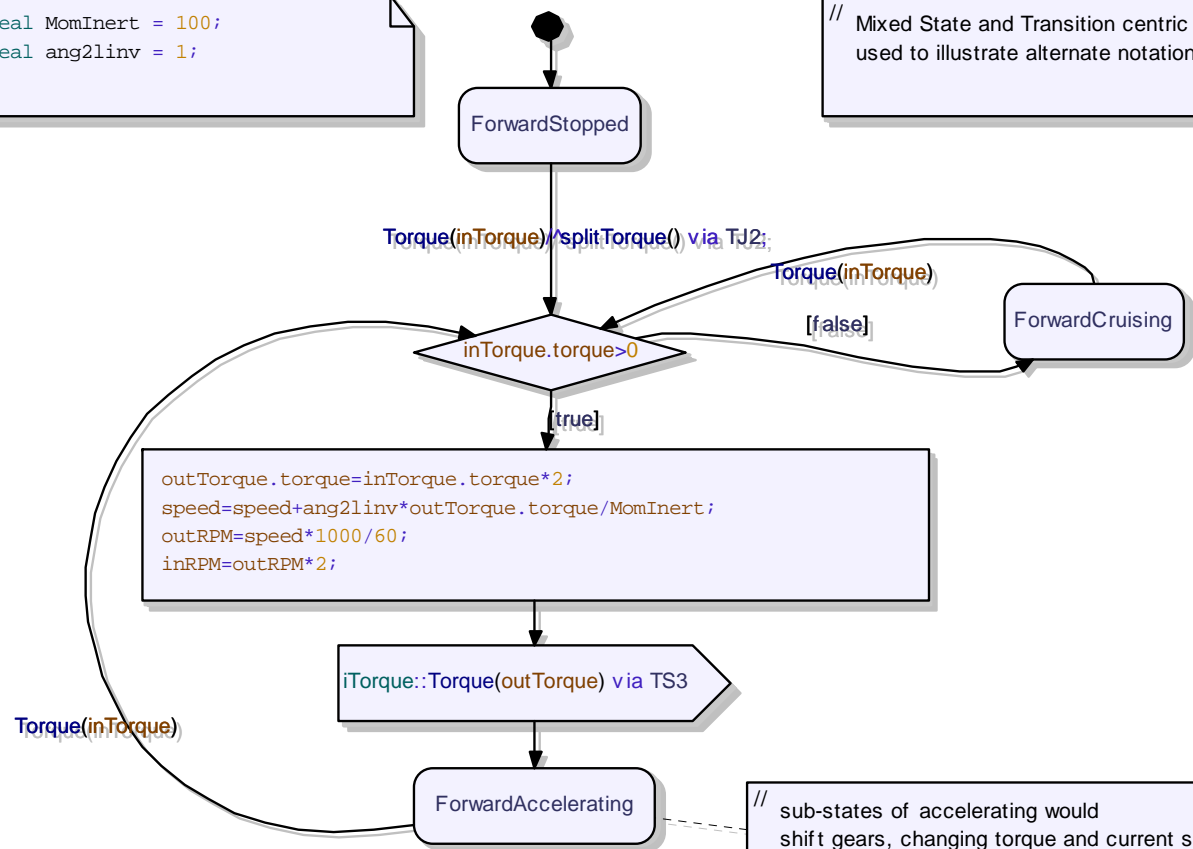


state Forward (1/2)

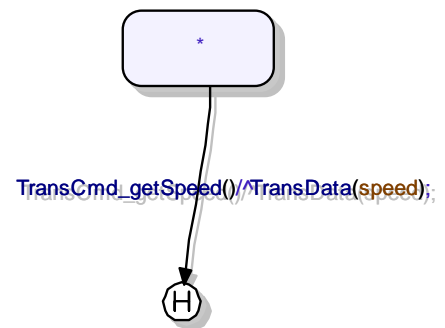
Amplify Torque

```
Real MomInert = 100;
Real ang2linv = 1;
```

// Mixed State and Transition centric state machine diagram used to illustrate alternate notation.



// sub-states of accelerating would shift gears, changing torque and current speed. Current assumption is constant gear ratio of 2. No drag, i.e. constant acceleration.

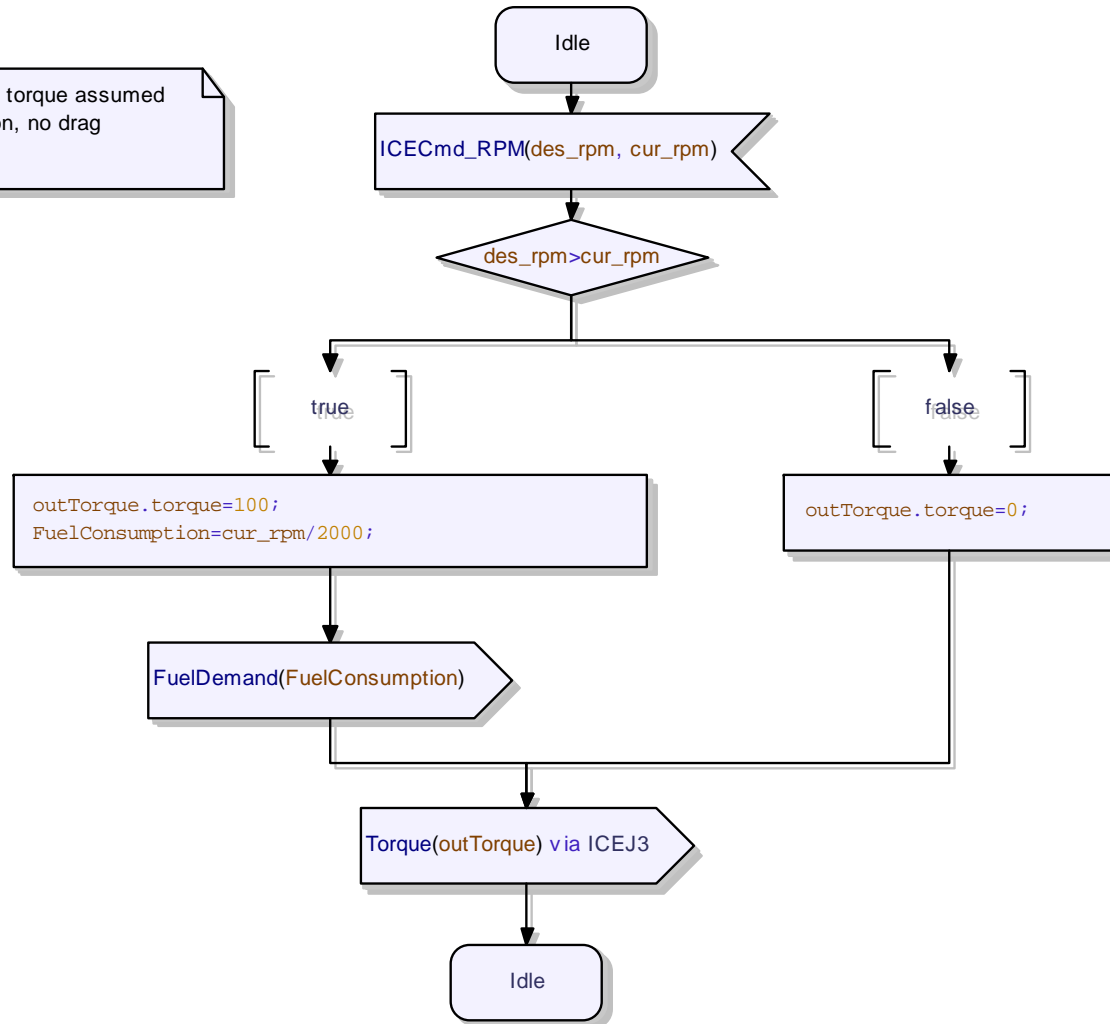


State Machine Diagram – Prod_Torque

Prod_Torque

stateMachine InternalCombustionEngine :: initialize. {2/2}

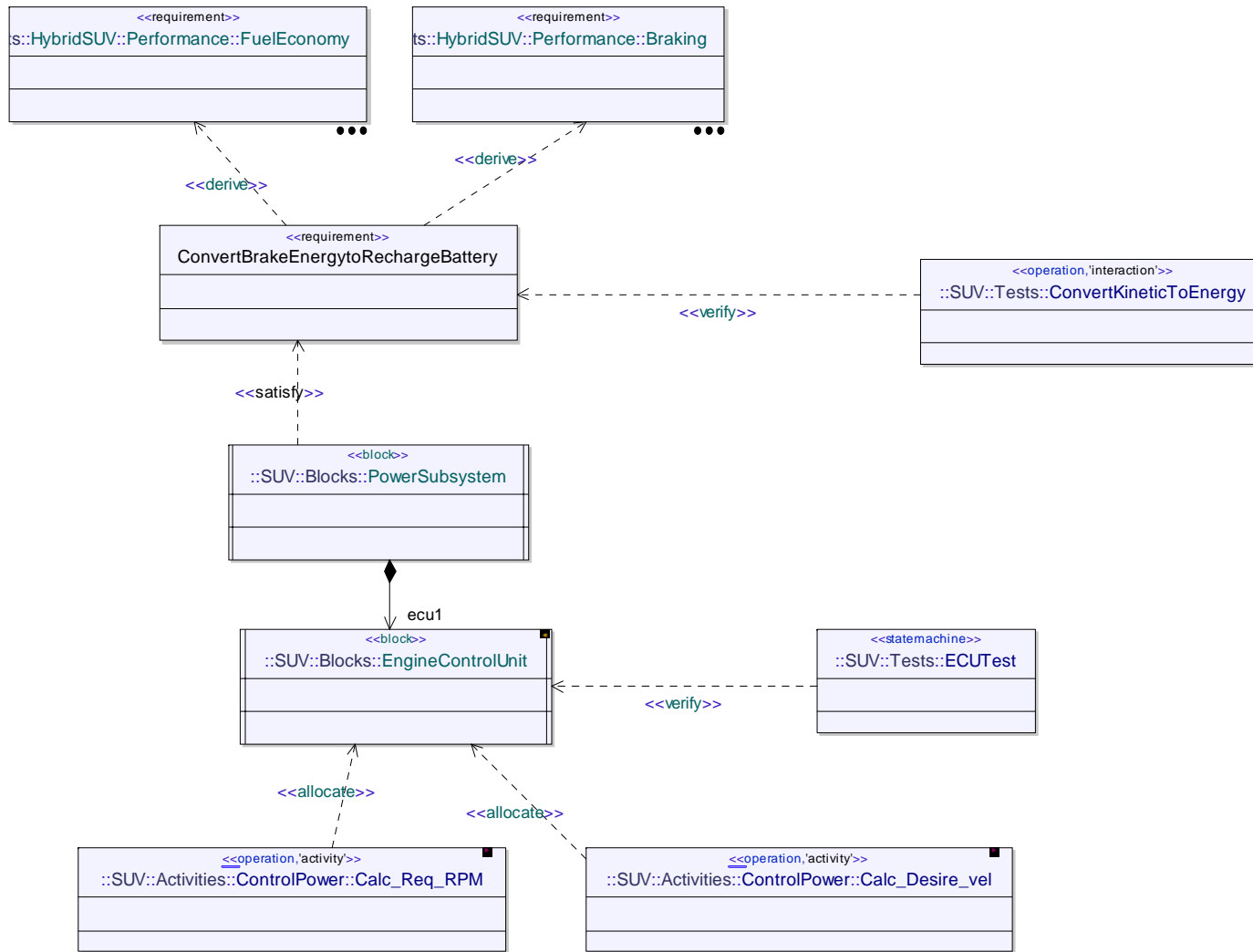
// Constant torque assumed
No friction, no drag



Traceability Across Lifecycle



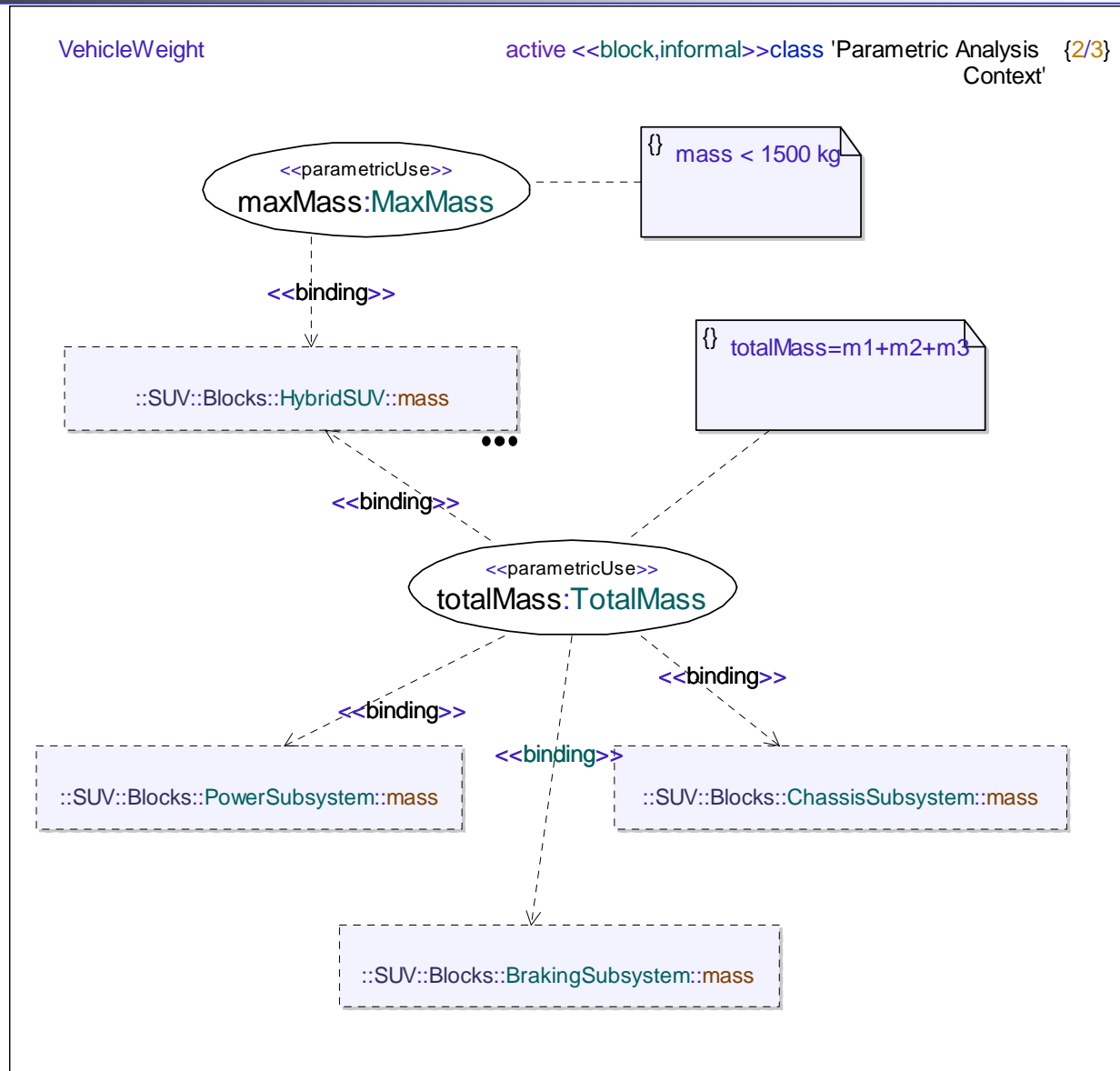
Requirement diagram 1 package Requirements {6/6}



Parametric Constraint Diagram

- Used to express constraints between quantifiable properties (aka non-functional characteristics) of assemblies
 - reusable
 - non-causal
- Defined as a stereotype
 - expression: text string specifies the constraint
 - expression language can be formal (e.g. MathML, OCL ...) or informal
 - computational engine is defined by applicable analysis tool and not by SysML
- Integral part of technical performance measures (roll-up) and trade studies (design decisions).

Parametric Constraint Usage



SysML Properties

- SysML Extension for Property to address
 - ValueType
 - Value (real or complex), Units, Dimensions, and Quantity
 - model library containing 39 pre-defined ValueTypes provided
 - easily extended by users to include additional types
 - Probability Distribution
 - distribution type, mean, variance
- New data types
 - Real
 - Complex



Wrap Up

Next Steps

- Support MDSD WG and OMG ADTF design reviews
- Close out actions from design reviews
- Iteratively, incremental update the specification
 - address issues identified by reviews
 - address issues identified by public feedback
- Publish a SysML v. 1.0 revision in January
- Update vendor implementations as required by the above
- Continue to actively pursue collaboration with other submission team to achieve single unified SysML submission

Summary



- SysML v. 1.0a is first complete version of SysML
 - smaller, more semantically expressive and executable
- SysML v. 1.0a addresses vast majority of SysML v. 0.9 issues
 - remainder will be addressed by SysML v. 1.0b
- SysML v. 1.0a addresses all INCOSE MDSD WG review recommendations with exception of HWCIs and CSCIs
 - HWCIs and CSCIs will be addressed by SysML v. 1.0b
- SysML v. 1.0a prototyping has provided valuable implementation and usability feedback
 - prototyping and beta-testing conducted May 2005- October 2005.
 - support for SysML v.1.0a available in TAU G2 v. 2.6; other tools TBA
- SysML v. 1.0a executable sample problem exceeds requirements and validates language and sample problem integrity
 - executable model capability identified as a valid Measure of Effectiveness by INCOSE MDSD WG review team
- SysML needs your support
 - provide timely review feedback
 - encourage convergence into unified SysML standard

Further Info

- SysML Open Source Project
 - www.SysML.org (includes FAQ page)
 - <mailto:SysMLforum@googlegroups.com>
- SysML Partners
 - www.SysML.org/partners.htm
 - <mailto:Partners@SysML.org>
- SysML v. 1.0a feedback
 - www.SysML.org/feedback.htm (feedback form)
 - <mailto:feedback@SysML.org>