SysML v2 Submission Team (SST)
A Look Ahead at SysML v2

SST

Wasatch INCOSE Chapter

June 11, 2020

Sanford Friedenthal
safriedenthal@gmail.com
Co-lead, SysML v2 Submission Team

Presentation Purpose

SST

- Share SysML v2 Submission Team (SST) approach
  - Background and motivation
  - What to expect from SysML v2
  - Contrast SysML v2 with SysML v1
  - Progress and plans

Systems Modeling Language™ (SysML®)

SST

Supports the specification, analysis, design, and verification and validation of complex systems that may include hardware, software, information, processes, personnel, and facilities

- SysML has evolved to address user and vendor needs
  - v1.0, adopted in 2006; v1.6, current version; v1.7, in process
- SysML has facilitated awareness and adoption of MBSE
- Much has been learned from using SysML for MBSE

SysML v2 Objectives

SST

Increase adoption and effectiveness of MBSE by enhancing…

- Precision and expressiveness of the language
- Consistency and integration among language concepts
- Interoperability with other engineering models and tools
- Usability by model developers and consumers
- Extensibility to support domain specific applications
- Migration path for SysML v1 users and implementors

SysML v2 Requests for Proposals *SST*

- SysML v2 RFP issued December, 2017
  - Initial Submission: August, 2020
  - Revised (Final) Submission: May, 2021
- SysML v2 API & Services RFP issued June, 2018
  - Initial Submission: August, 2020
  - Revised (Final) Submission: May, 2021
- SysML v2 Submission Team (SST) formed December 2017
  - Leads: Sandy Friedenthal, Ed Seidewitz

*Initial and revised **submission** dates reflect extensions accepted by OMG*

SysML v2 Submission Team (SST) *SST*

- A broad team of end users, vendors, academics, and government liaisons
  - Over 100 members representing 65+ organizations
- Developing submissions to both RFPs
- Driven by RFP requirements and user needs

SST Participating Organizations

SST

Academia/Research
End User

Tool Vendors
Government Rep

INCOSE rep *

- Aerospace Corp
- Airbus
- ANSYS medini
- Aras
- Army Aviation & Missile Center
- Army Office of Chief SE
- BAE
- BigLever Software
- Boeing
- Army CCDC Armaments Center
- CEA
- Contact Software
- DEKonsult
- Draper Lab
- Elbit Systems of America
- ESTACA
- Ford
- Fraunhofer FOKUS
- General Motors
- George Mason University
- GfSE
- Georgia Tech/GTRI
- IBM
- Idaho National Laboratory
- IncQuery Labs
- Intercax
- Itemis
- Jet Propulsion Lab
- John Deere
- Kenntnis
- KTH Royal Institute of Technology
- LieberLieber
- Lightstreet Consulting
- Lockheed Martin
- MathWorks
- Maplesoft
- Mgnite Inc
- MITRE
- ModelAlchemy Consulting
- Model Driven Solutions
- Model Foundry
- NIST
- No Magic/Dassault Systemes
- OAR
- Obeo
- OOSE
- Ostfold University College
- Phoenix Integration
- PTC
- Qualtech Systems, Inc (QSI)
- Raytheon
- Rolls Royce
- SAF Consulting *
- SAIC
- Siemens
- Sierra Nevada Corporation
- Simula
- Sodius Willert
- System Strategy *
- Tata Consultancy Services
- Thales
- Thematix
- Tom Sawyer
- UFRPE
- University of Cantabria
- University of Alabama in Huntsville
- University of Detroit Mercy
- University of Kaiserslautern / VPE
- Vera C. Rubin Observatory
- Vitech
- 88solutions

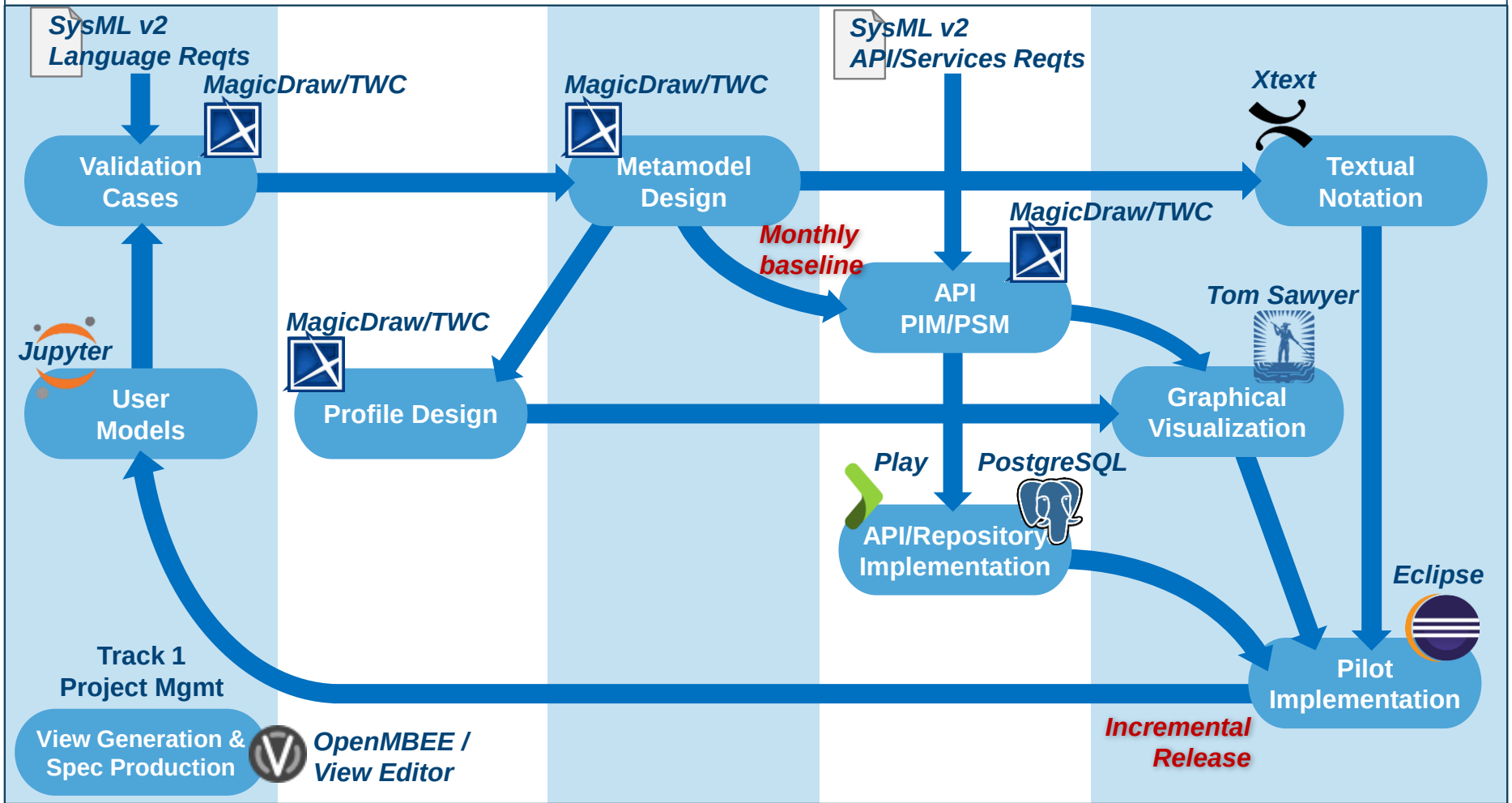
SST Tracks / Leads

SST

1. Project Management – Ed Seidewitz, Sandy Friedenthal
   - Infrastructure – John Watson, Chris Delp
2. Requirements V&V – Sandy Friedenthal
3. Profile Development – Yves Bernard, Tim Weilkiens
4. Metamodel Development – Chas Galey, Karen Ryan
5. API/Services Development – Manas Bajaj
6. Pilot Implementation – Ed Seidewitz

SST Incremental Approach

SST



SysML v2 Validation Cases

SST

- The following 16 validation cases capture initial required language functionality

  Reflects 2/3 of the SysML v2 RFP requirements

  - 1-Parts Tree
  - 2-Parts Interconnection
  - 3-Function-based Behavior
  - 4-Functional Allocation
  - 5-State-based Behavior
  - 6-Individuals and Snapshots
  - 7-Variant Configuration
  - 8-Requirements
  - 9-Verification
  - 10-Analysis and Trades
  - 11-View and Viewpoint
  - 12-Dependency Relationships
  - 13-Model Containment
  - 14-Language Extension
  - 15-Properties, Values, & Expressions
  - 16-Proxy validation case

Current preliminary design baseline and pilot implementation

In work

Key Elements of SysML v2

SST

- New Metamodel that is not constrained by UML
  - Grounded in formal semantics
- Robust visualizations based on flexible view & viewpoint specification and execution
  - Graphical, Tabular, Textual
- Standardized API to access the model

Usage Focused Modeling Approach

SST

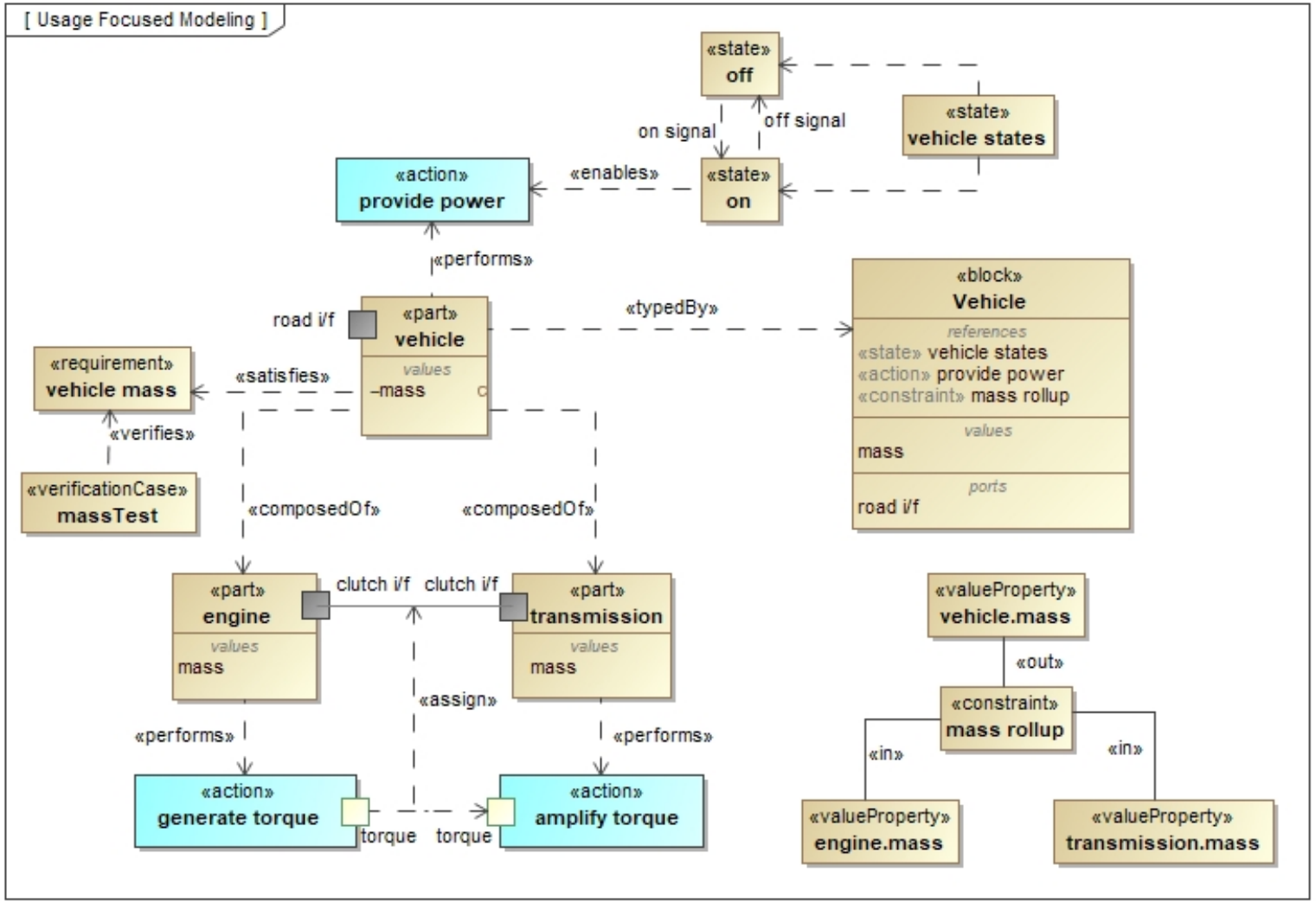
A paradigm shift to make SysML v2 more precise and more intuitive to use

- Emphasizes modeling of *usages (e.g., parts on an ibd)*
  - Decompose, connect, relate, and group usages
- Supports other language requirements
  - variant design configurations, individuals, …

Usage Focused Modeling Approach
Multiple Views of a System

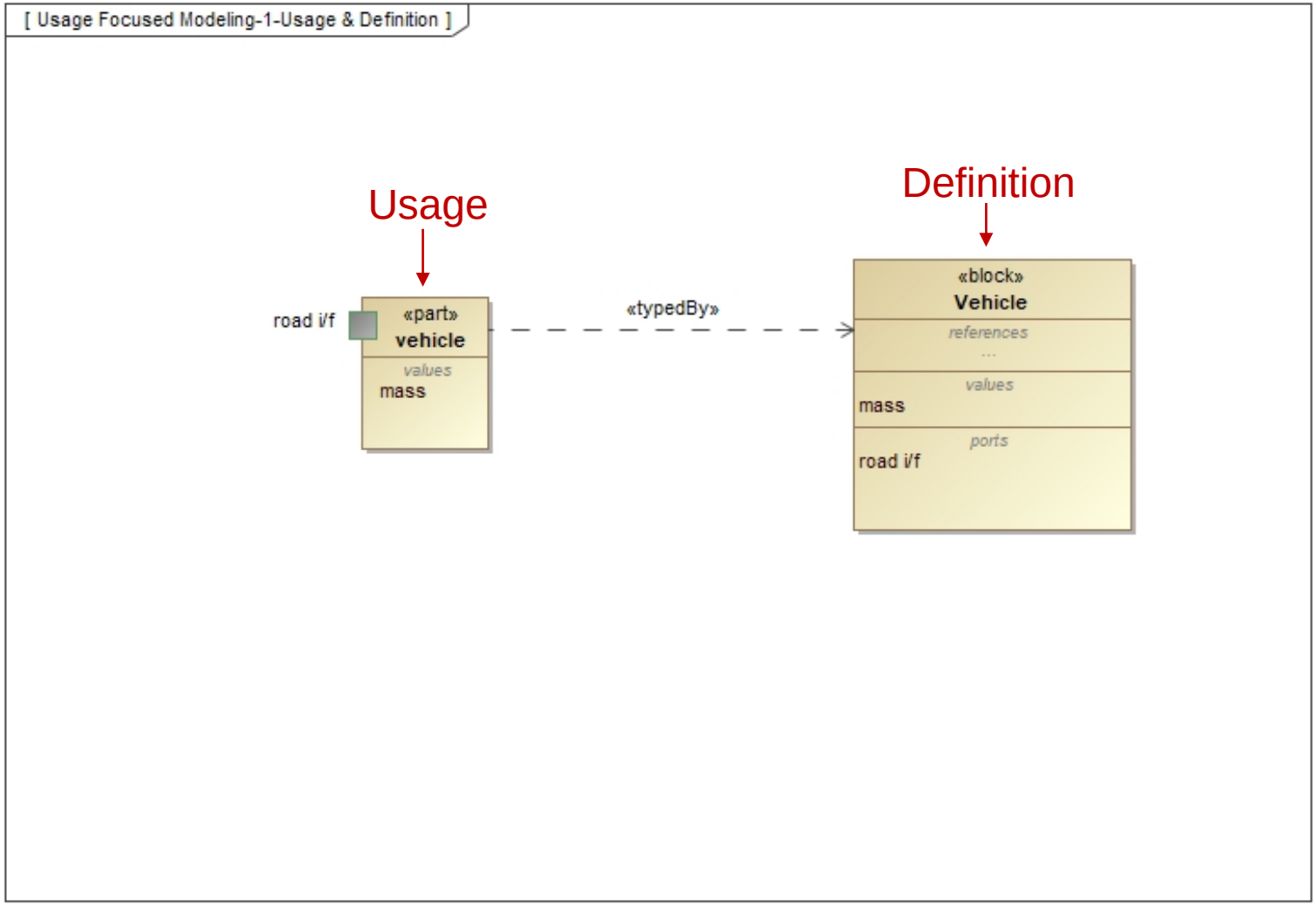
SST

Graphical notation for illustrative purposes only



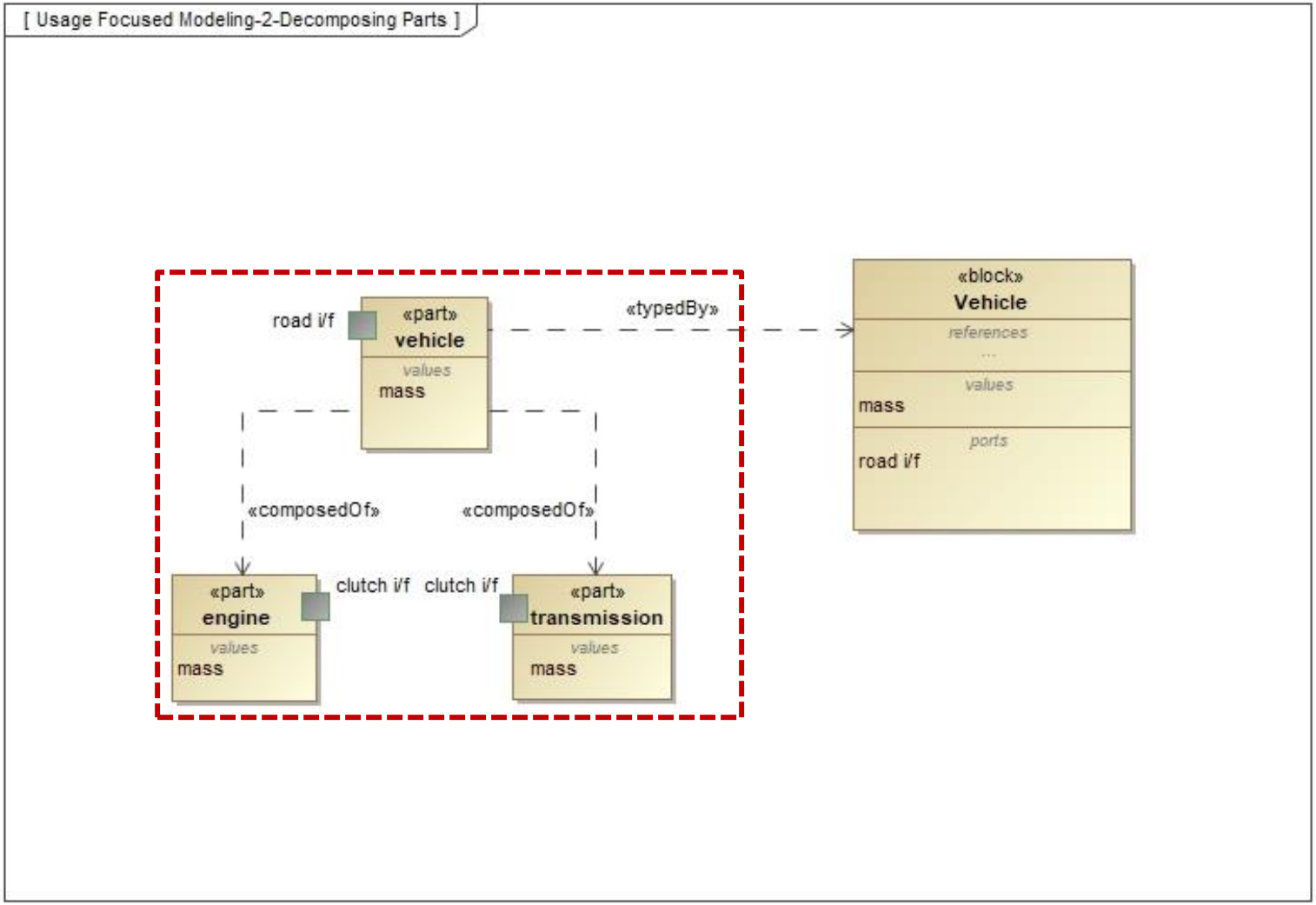
Usage Focused Modeling Approach
Usage & Definition

SST



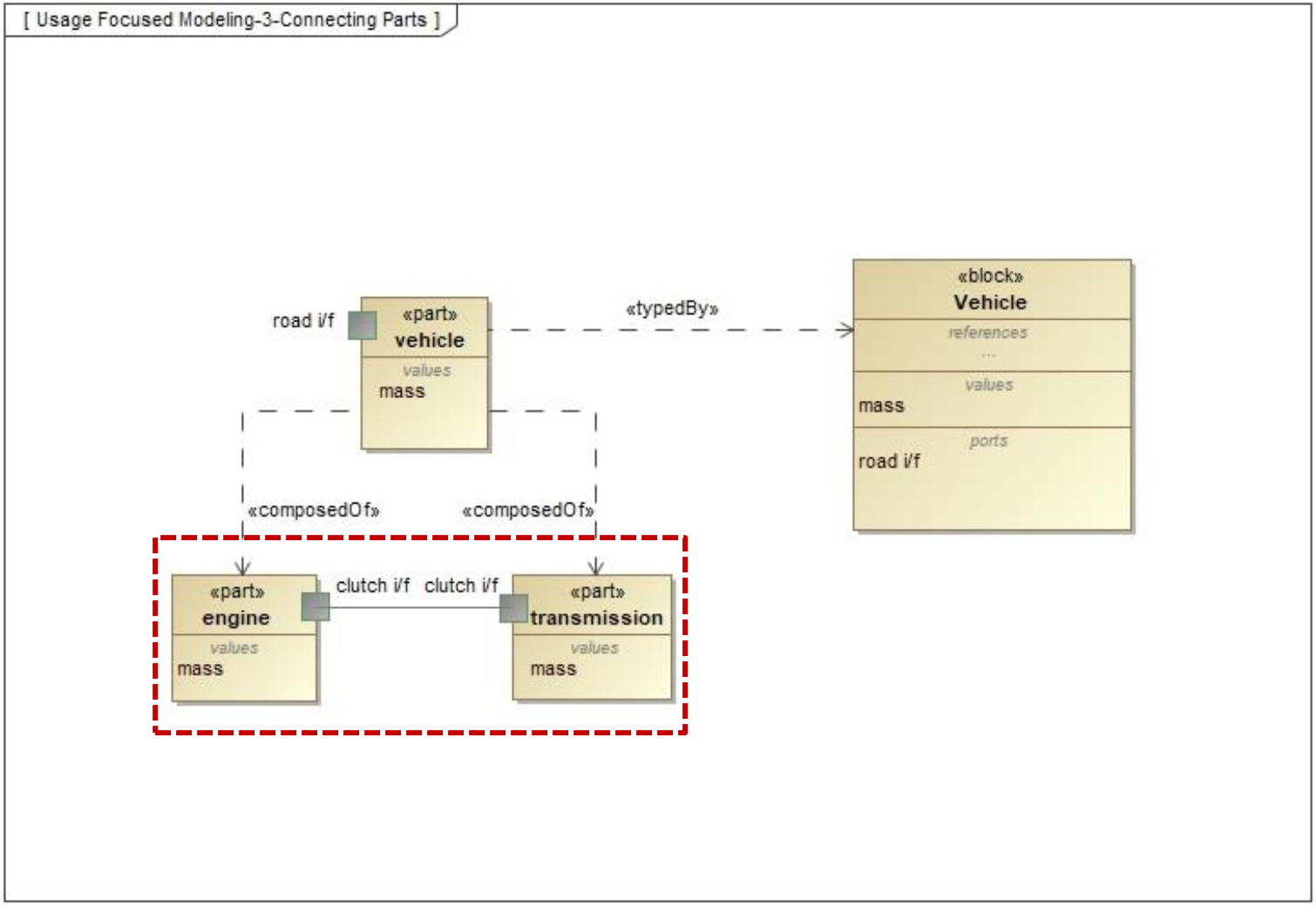
Usage Focused Modeling Approach
Decomposing Parts

SST



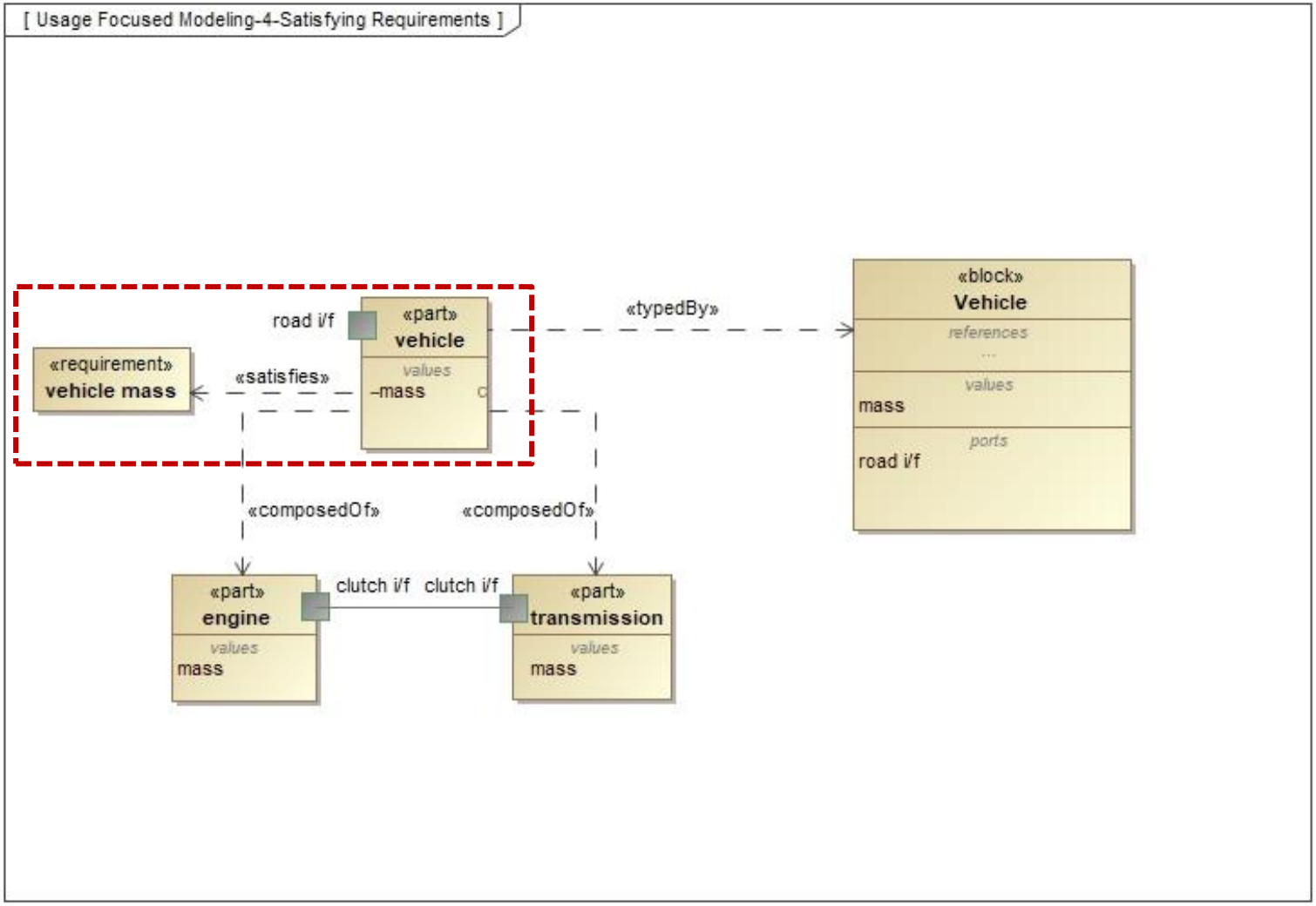
Usage Focused Modeling Approach
Connecting Parts

SST



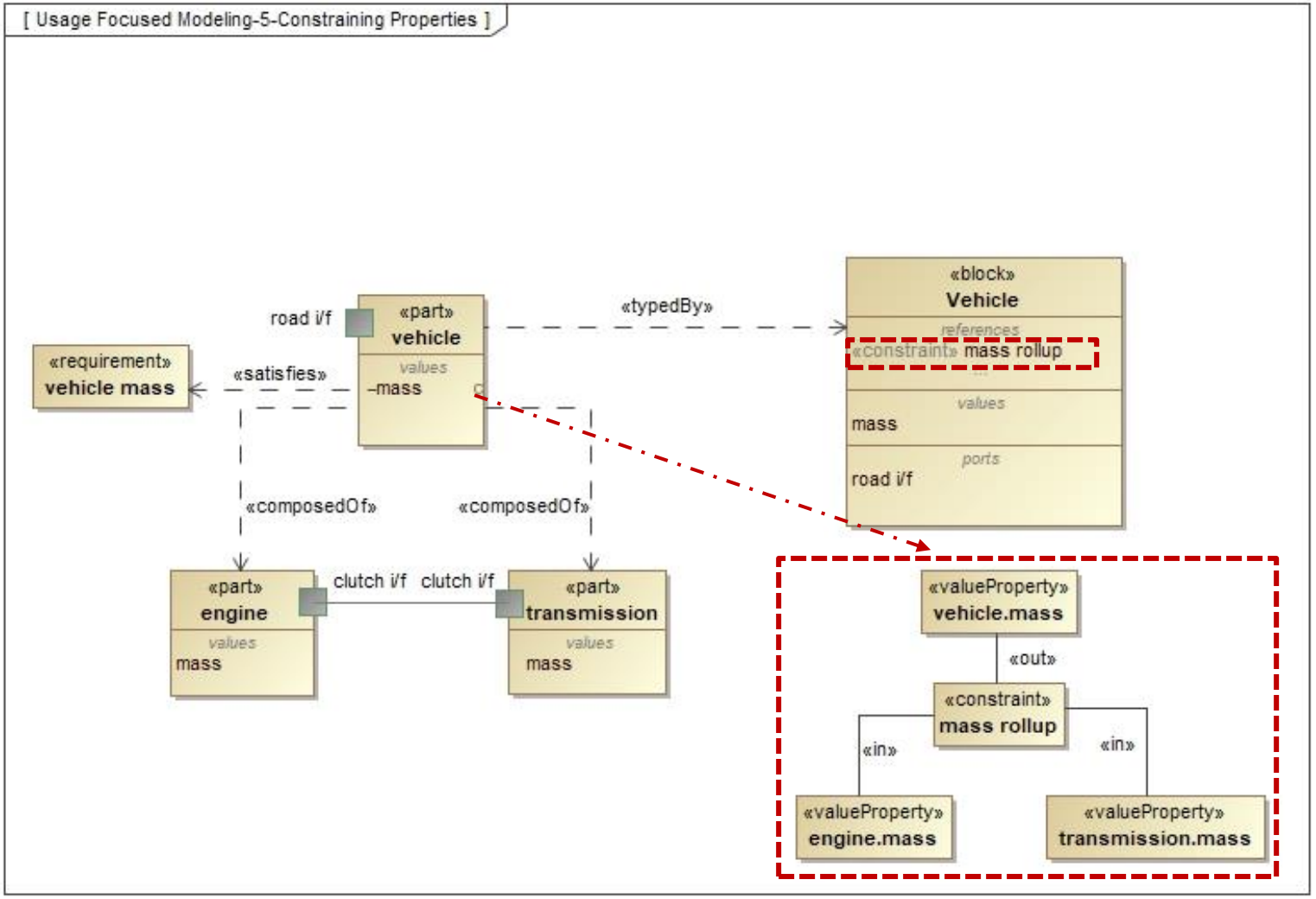
Usage Focused Modeling Approach Satisfying Requirements

SST



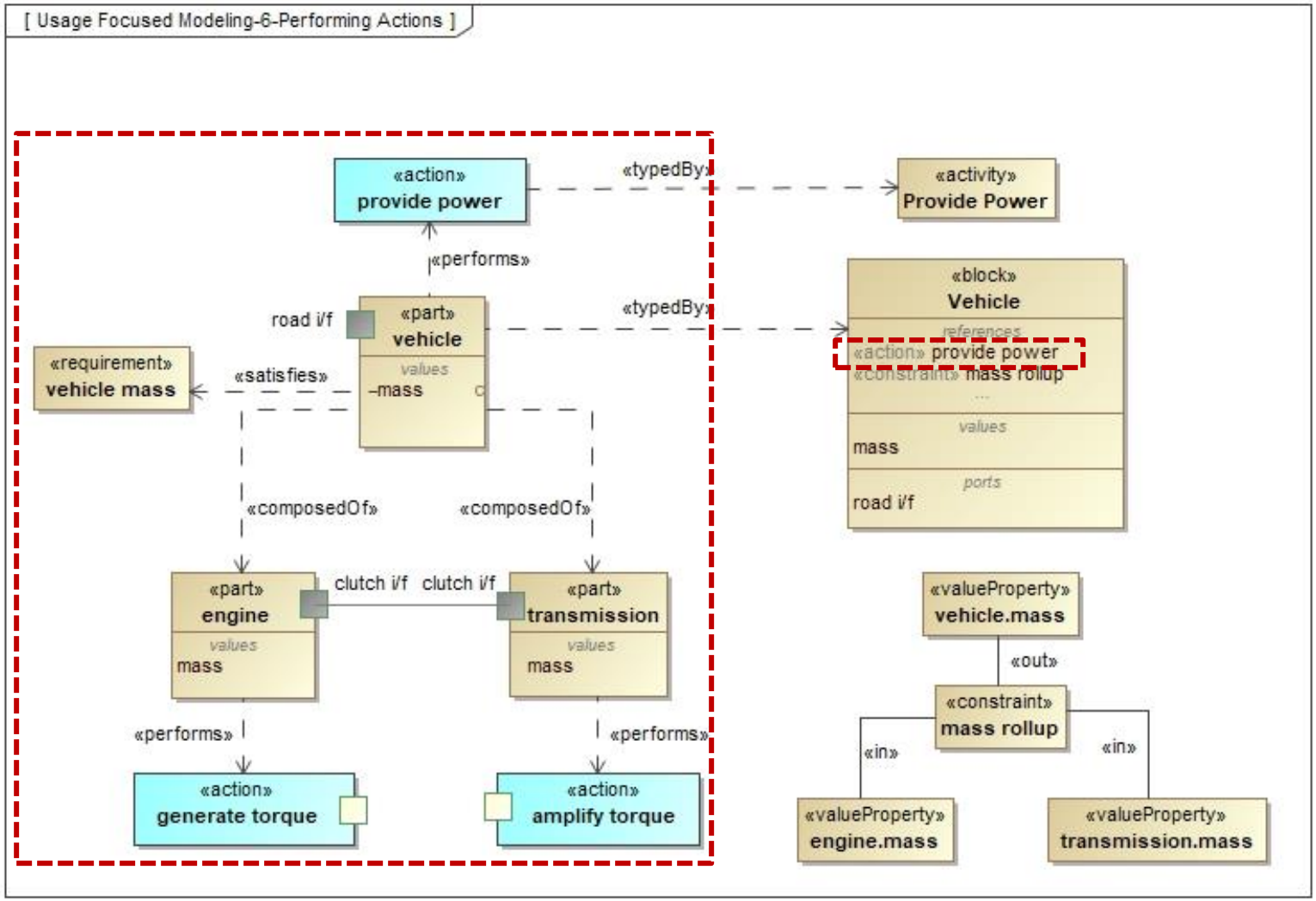
Usage Focused Modeling Approach
Constraining Properties

SST



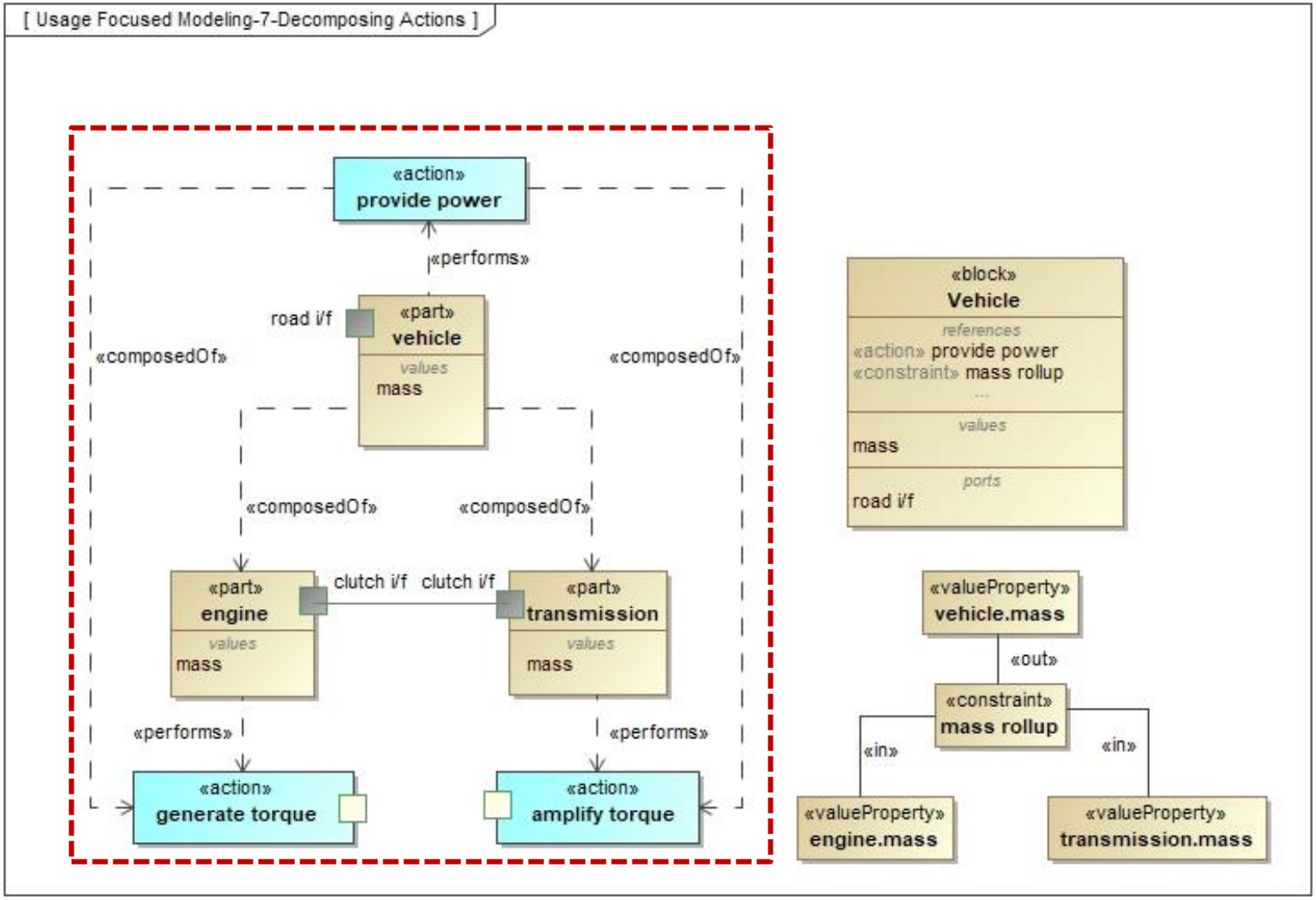
Usage Focused Modeling Approach
Performing Actions

SST



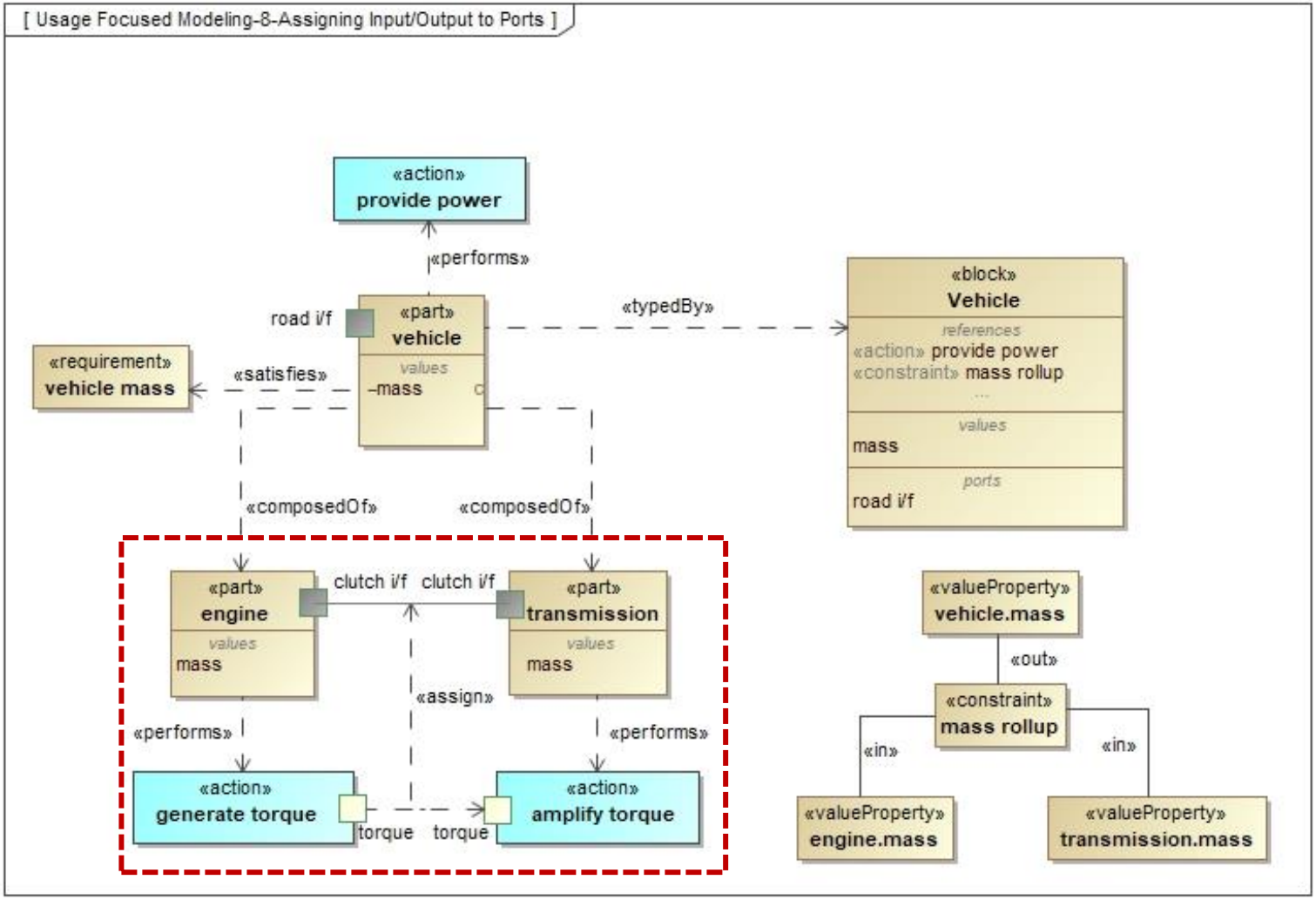
Usage Focused Modeling Approach
Decomposing Actions

SST



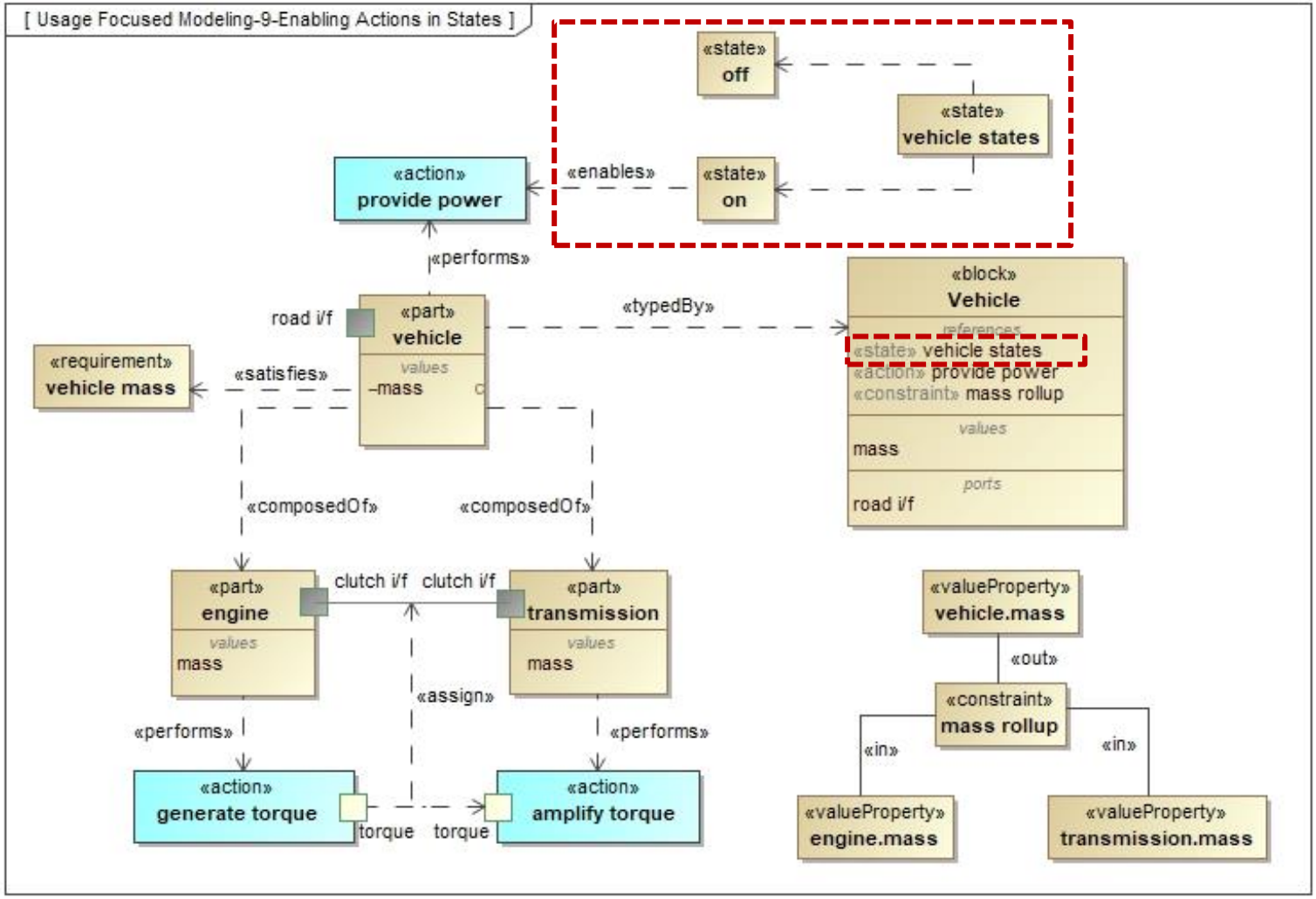
Usage Focused Modeling Approach
Assigning Input/Output to Ports

SST



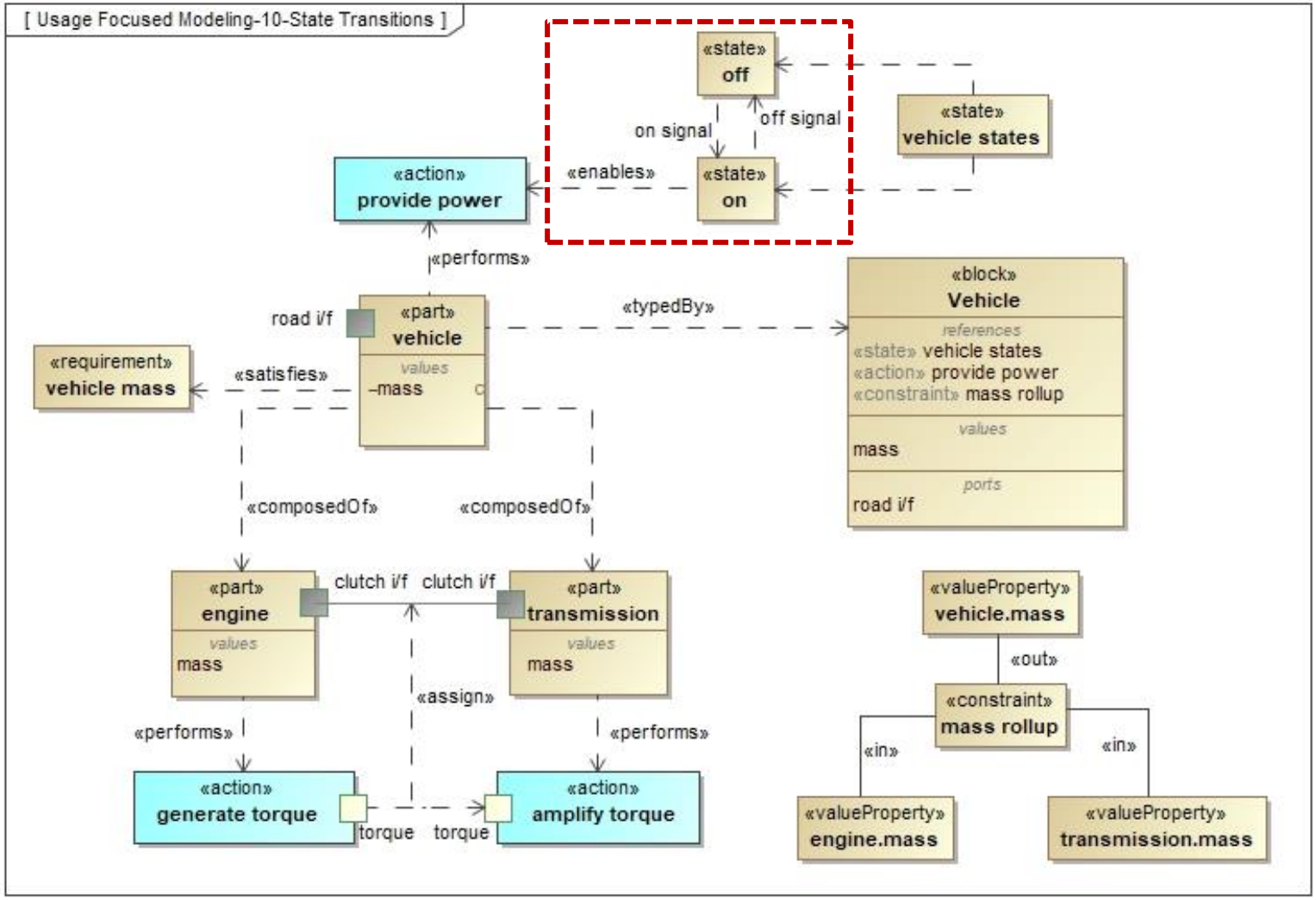
Usage Focused Modeling Approach
Enabling Actions in States

SST



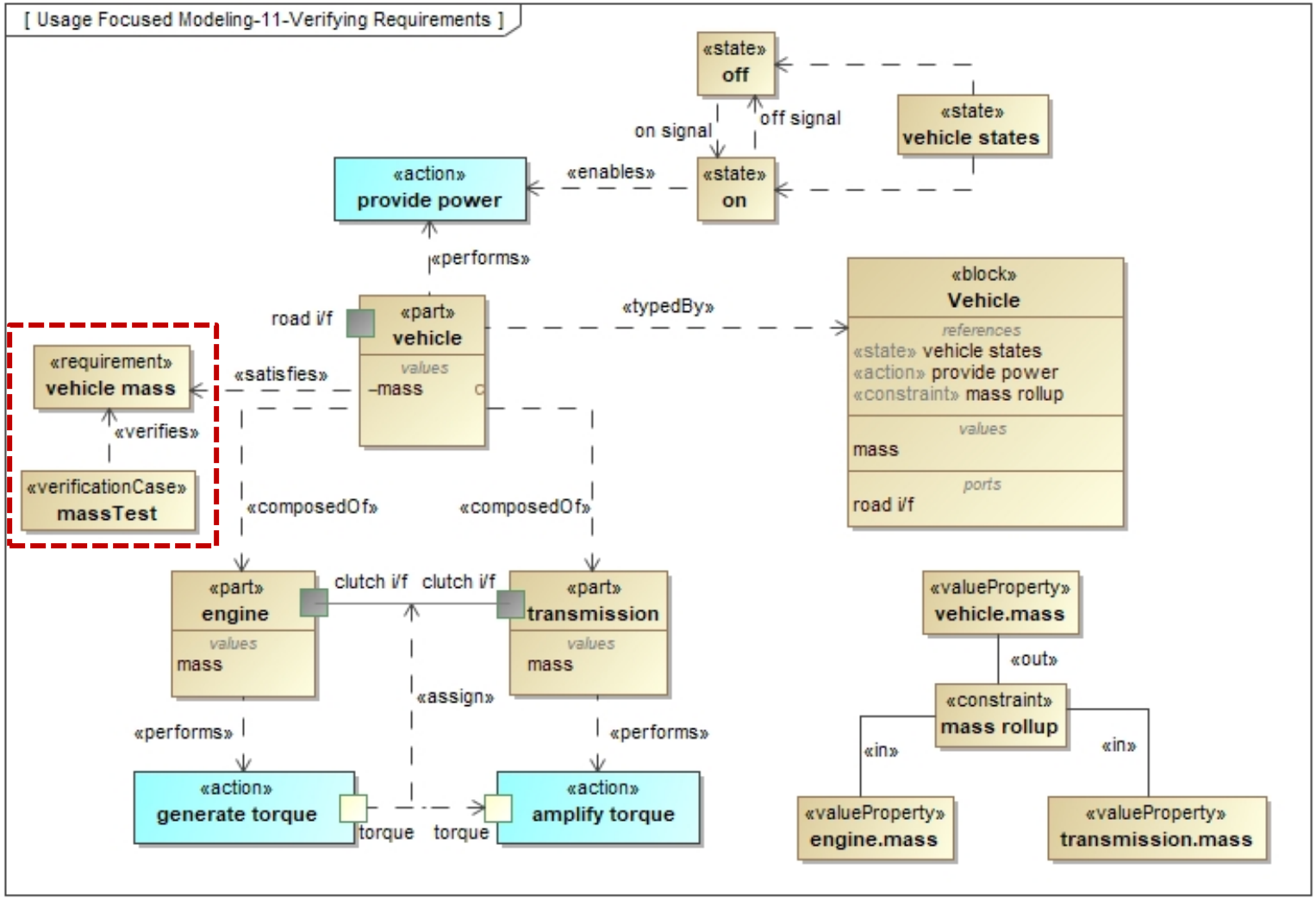
Usage Focused Modeling Approach
State Transitions

SST



Usage Focused Modeling Approach
Verifying Requirements

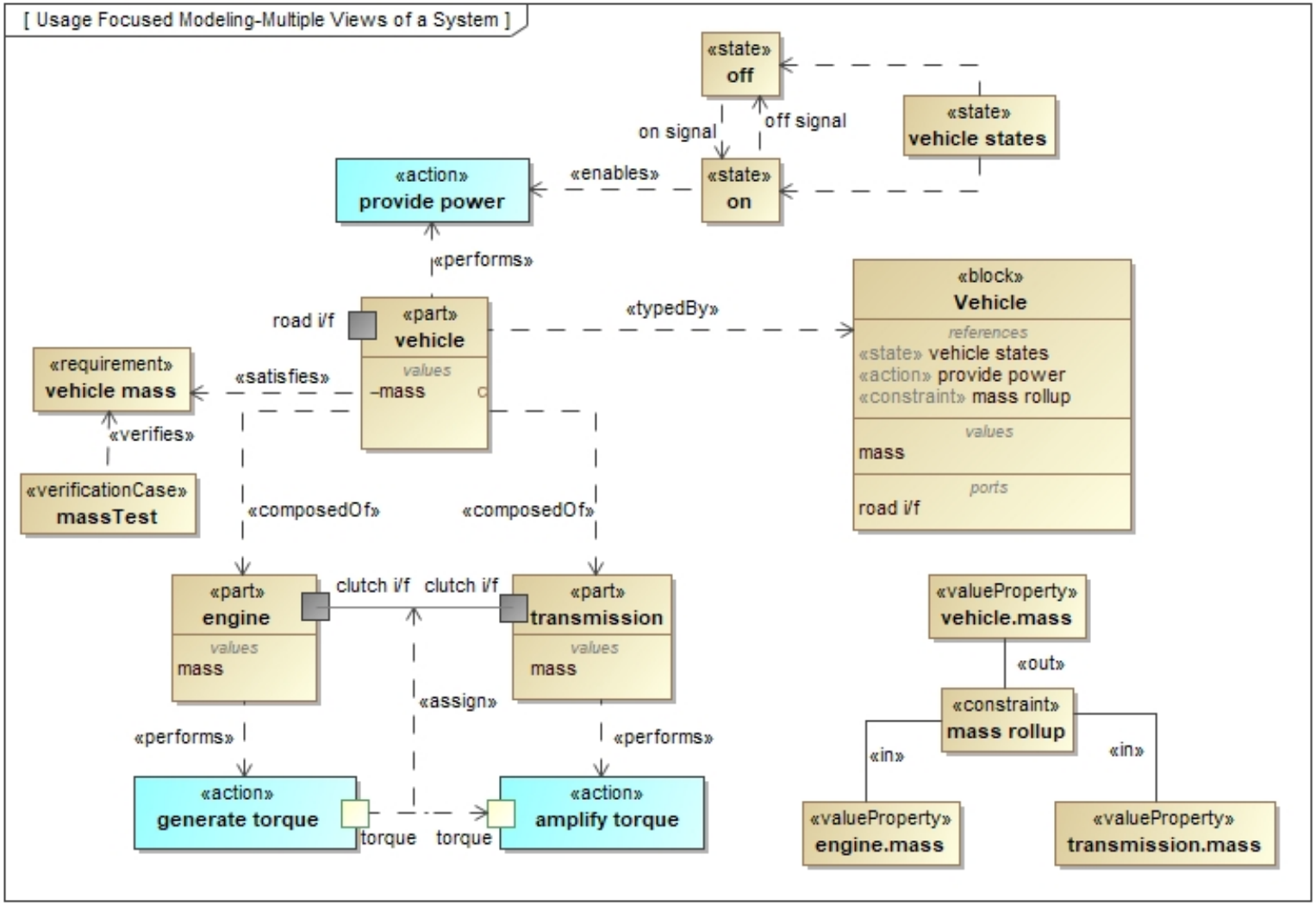
SST



Usage Focused Modeling Approach
Multiple Views of a System

SST

Graphical notation for illustrative purposes only



Example Using Textual Notation Definitions

SST

```
package sfriedenthal_VehicleModel_1{
   package Definitions{
      package PartDefinitions{
         block Vehicle {
            value mass :> ISQ::mass;
         }
         block Engine;
         block Cylinder;
         block Transmission;
      }
      package PortDefinitions{
         port def FuelCmdPort;
         port def VehicleToRoadPort;
      }
      package ActionDefinitions{
         activity ProvidePower (
            in fuelCmd:FuelCmd,
            out wheelToRoadTorque:Torque[2]
         );
      package StateDefinitions {
         state def VehicleStates;
         state def ControllerStates;
      }
      package ValueDefinitions{
         import ScalarValues::*;
      }
   }
```

Some simplifications have been made for the purposes of presentation

Example Using Textual Notation Usages/Configuration_a

SST

```
package VehicleConfigurations{
    import Definitions::*;
    package VehicleConfiguration_a{
        package VehiclePartsTree{
            part vehicle_a:Vehicle{
                value mass redefines mass=1750;
                part frontAxleAssembly:AxleAssembly{
                    part frontAxle:Axle;
                    part frontWheels:Wheel[2];
                }
                part rearAxleAssembly:AxleAssembly{
                    part rearAxle:Axle;
                    part rearWheels:Wheel[2];
                }
            }
        }
    }
```

vehicle_a is typed by Vehicle

Some simplifications have been made for the purposes of presentation

Example Using Textual Notation Usages/Configuration_b

SST

```
package VehicleConfiguration_b{
    import VehicleConfiguration_a::*;
    package VehiclePartsTree{
        part vehicle_b :> vehicle_a{
            value mass redefines vehicle_a::mass=2000;
            port fuelCmdPort:FuelCmdPort;
            port vehicleToRoadPort:VehicleToRoadPort{
                port wheelToRoadPort1:WheelToRoadPort;
                port wheelToRoadPort2:WheelToRoadPort;
            }
            perform VehicleActionTree::providePower;
            exhibit States::vehicleStates;
```

vehicle_b is a kind of vehicle_a

vehicle_b value property

vehicle_b port

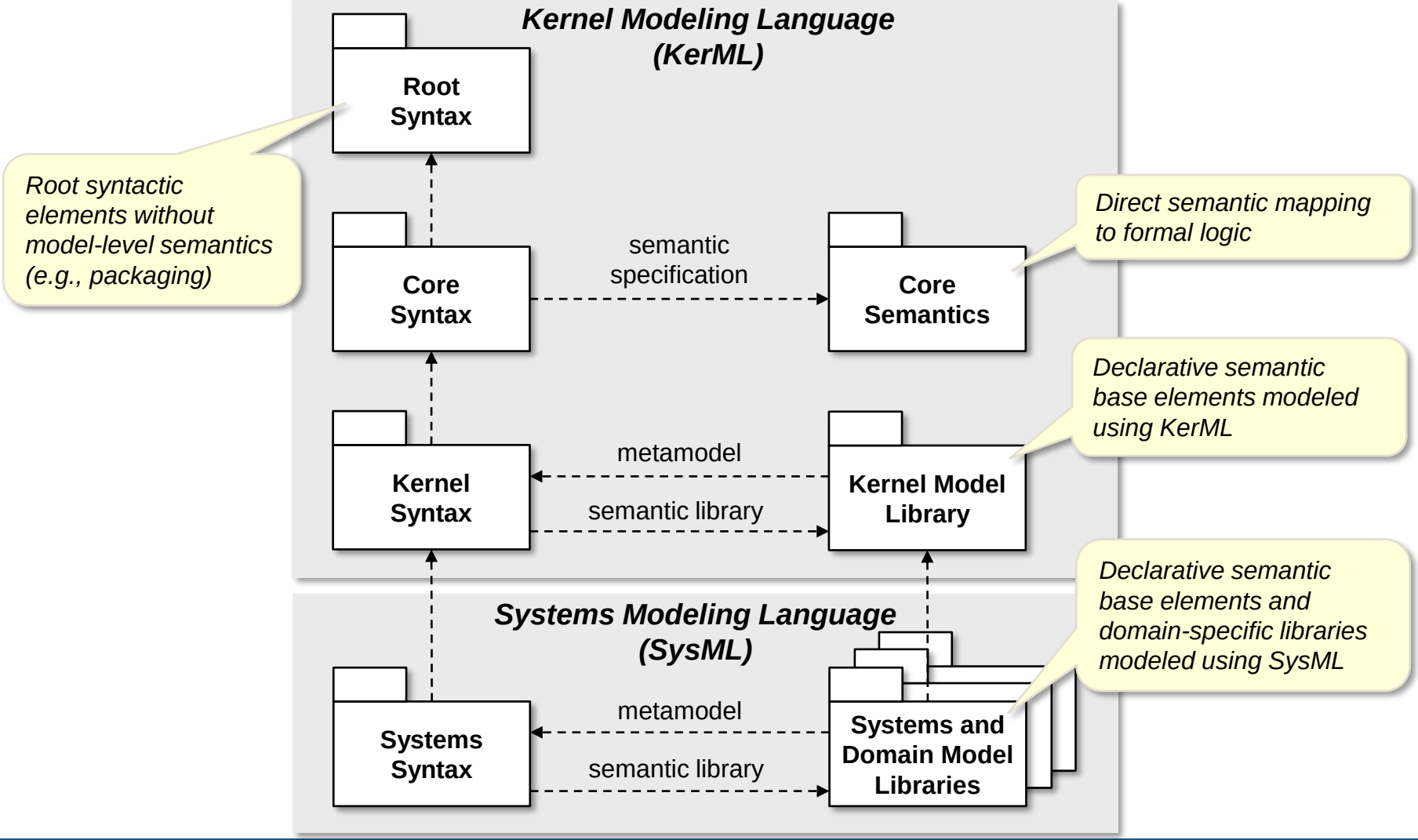
vehicle_b function

vehicle_b states

Some simplifications have been made for the purposes of presentation

SysML v2 Language Architecture

SysML v2 Language Architecture SST



SysML v2 API & Services

11 June 2020

SysML v2 API & Services

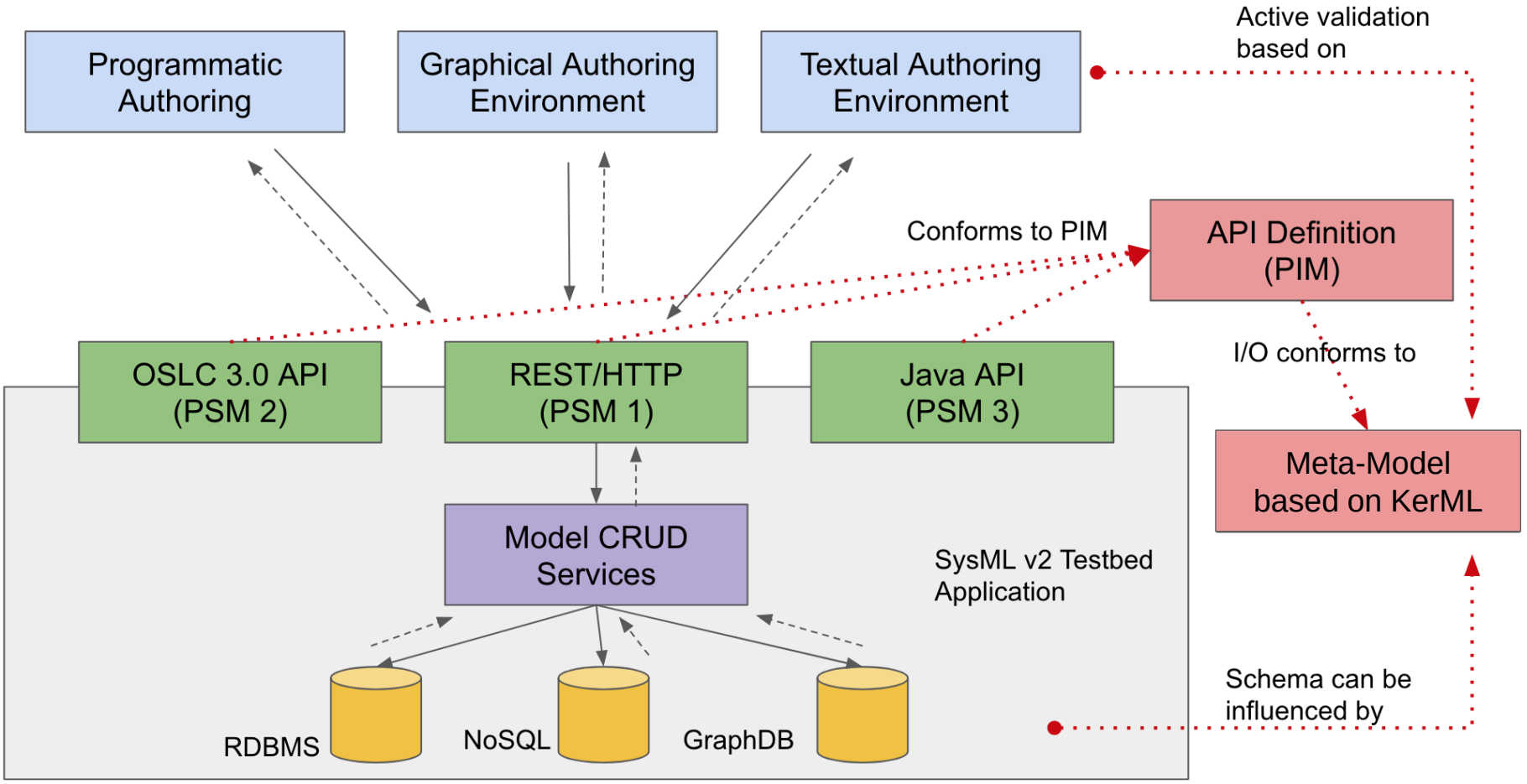
SST

- Enables other tools and applications to access SysML models in a standard way
- Provides services to:
  - Create, update, and delete elements
  - Query and navigate model
  - Other services including support for model management, analysis, view generation, transformation, and file export generation
- Facilitates use of different implementation technologies such as Rest, Java, and OSLC

Pilot Implementation Using Standard API

SST

High-Level Architecture of SysML v2 Testbed



Summary

Next Public Incremental Release (2020-03 release)

SST

- Publicly available on Google Drive
- Google group for comments and questions
- Content
  - Read me file (includes installation instructions)
  - Specification documentation (Parts 1, 2, 3)
  - Training material for SysML textual notation
  - Installation file for Jupyter tooling
  - Installation site for Eclipse plug-in
  - Web access to Tom Sawyer tooling/repository

Summary

SST

- SST is addressing RFP requirements and issues associated with SysML v1 to improve adoption and effectiveness
  - Precision and expressiveness
  - Consistency and integration among language concepts
  - Interoperability with other engineering models and tools
  - Usability by model developers and consumers
- Initial approach
  - SysML v2 metamodel that overcomes fundamental UML limitations
  - Flexible graphical notations and textual notation
  - Formal semantics
  - Standardized API for interoperability
- Working towards initial submission

Thank you!!