Learn a Simple MBSE Process by Connecting the Dots Using SysML

Tutorial Track B, T-4

Loyd Baker – VP of Technology

3SL, Inc

March 31, 2016

The presenter,

Loyd Baker, is VP for Technology with 3SL Inc., with extensive experience teaching automated MBSE methods to major US corporations and government agencies such as NASA, FAA, and DoE.

Provides training for Cradle (**https://www.threesl.com*)*** the systems engineering tool selected by NASA as their primary requirements management tool and used on several MBSE projects.

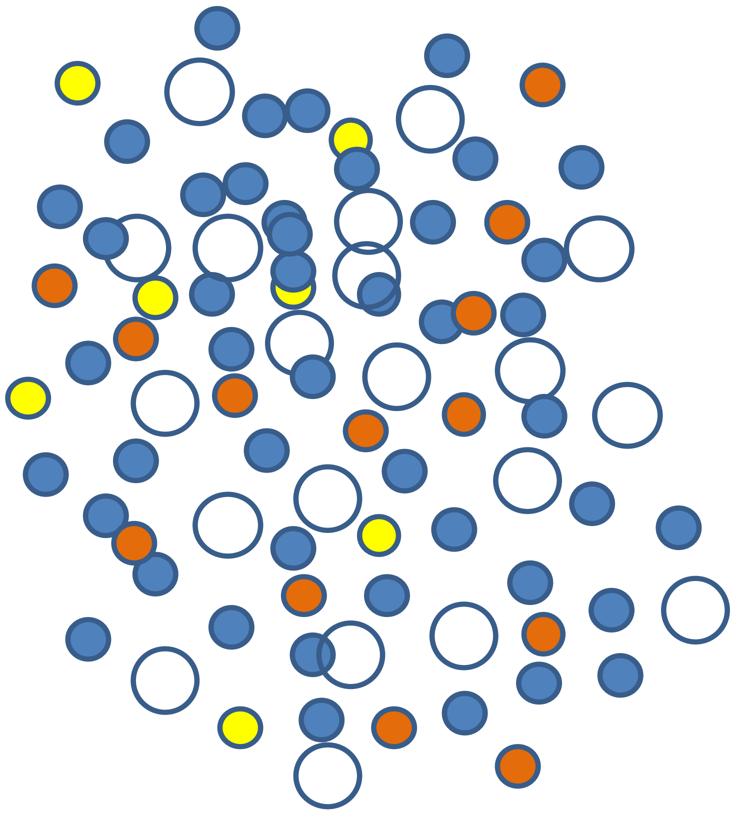
Past president of the Huntsville Chapter of International Council on Systems Engineering (INCOSE)

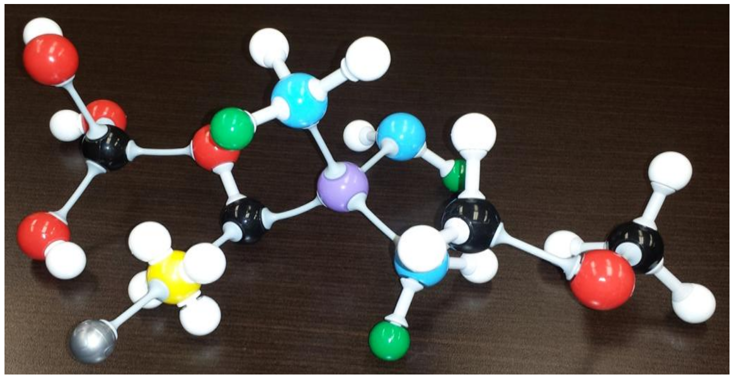
NASA Silver Snoopy Award winner for support of the Apollo missions, including Apollo 13.

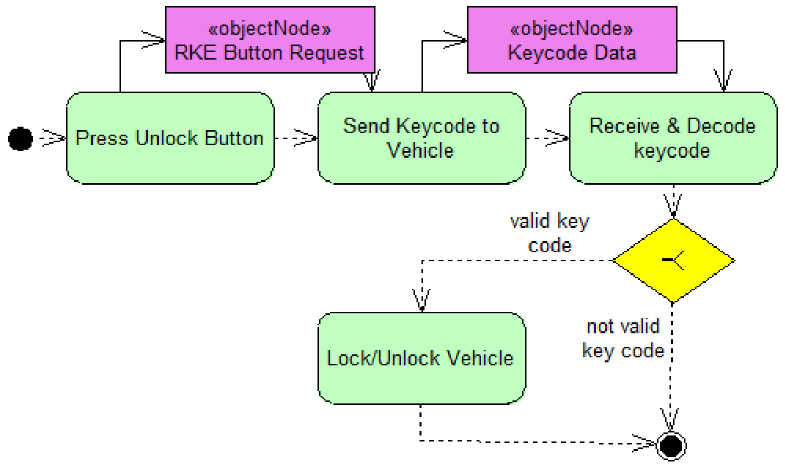
Model-Based Systems Engineering (MBSE) Defined

MBSE is all about Connecting the Dots to Tell a Story

A Piece of Information by itself has limited value, but when associated with one or more other pieces of information via relationships (i.e., cross reference links), the information has more value to the project







MBSE Support Capabilities Needed

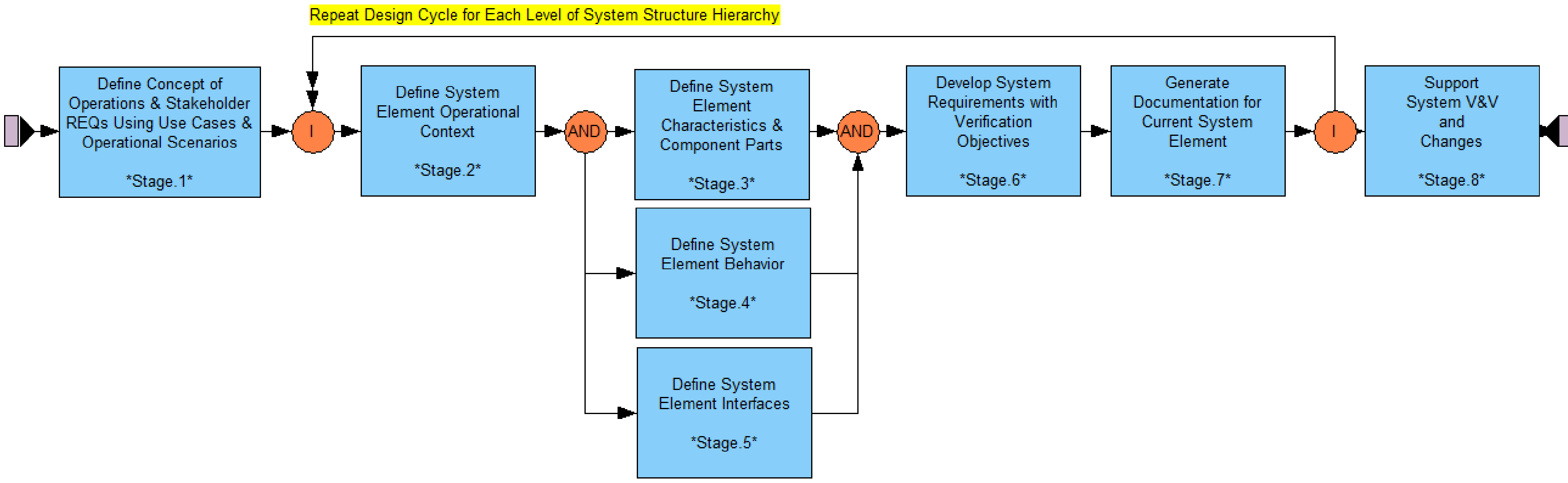
1. Documented Systems Engineering Processes - A documented Systems Engineering Process is needed to identify what information should be captured in order to develop a system/product architecture specification with traceability to the requirements.
2. Information Modeling Language – An Information Modeling Language is needed to capture information in a structured way so people and the automated tools can navigate through the Information Model and answer questions or return information for analysis or document generation.
3. Information-Model Repository – An Information Repository able to store and management multiple Information-Models and variants of a model is required. A model may consist of unique project information and information shareable by any project.
4. Automated Tools - A set of tools that can interact with the Information Repository and automate the capture and manipulation of information in the Information Repository, perform analyses, generate metrics, produce reports, generate documents using common templates, and produce XML/XMI exchange files for sharing information with detailed design tools.
5. Training - Classroom training on the MBSE Process and Methods and libraries of Example Models

MBSE – System Engineering Process

The MBSE Process activities provide the systems engineering work flow for the project and identify what modeling language information element-types and diagram-types will be needed and the automated tools that are needed to access the Information-Model in the Repository.

The following example process model diagram decomposes into many other diagrams providing more detail and every box has a detailed textual description but only the top level activities are shown in this presentation.

This process was used successfully on a NASA project using the Cradle Systems Engineering tool


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

- Circle with embedded I symbol : Two Iterate Nodes where everything between the nodes repeats
- Circle with embedded AND: Each exit branch from leading node indicates parallel processing

3SL

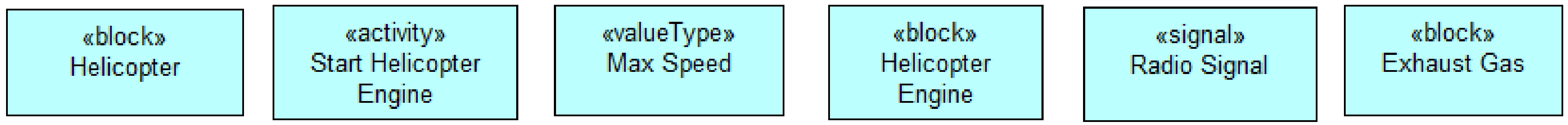
2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

MBSE - Information Modeling Language

A modeling language consists of **elements** (project information), the **properties** of the elements (e.g., description, rationale, etc), the **relationships** that connect the different elements together, and **diagrams** used to model concepts, logical behaviors, and system structures.

A graphical modeling language for systems engineers was developed by the OMG, INCOSE, and AP233 organizations. The language is know as SysML (System Modeling Language). It is being adopted by many companies and organizations.

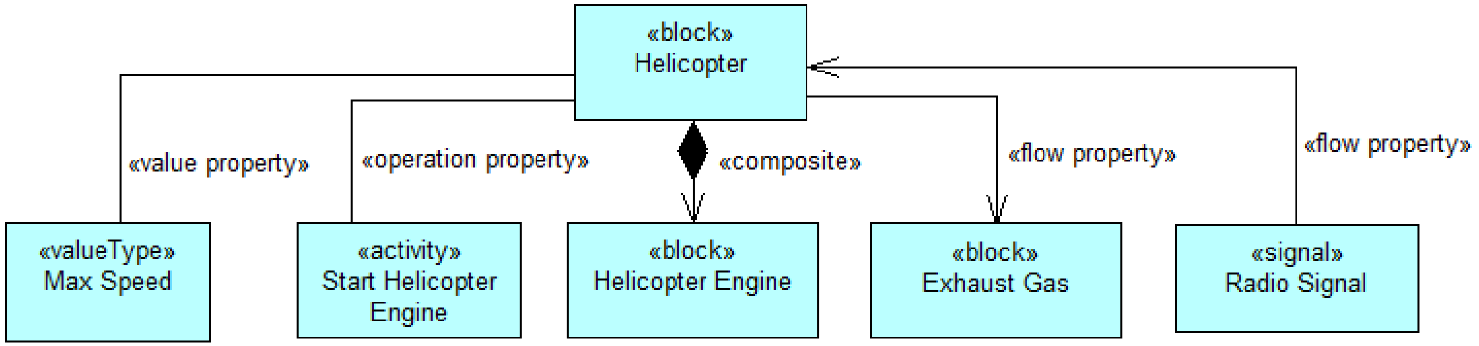
In SysML the different element-types used to specify a system or process are called **stereotypes**. The stereotype names are enclosed in « » symbols. In the following example shows five different element-types.



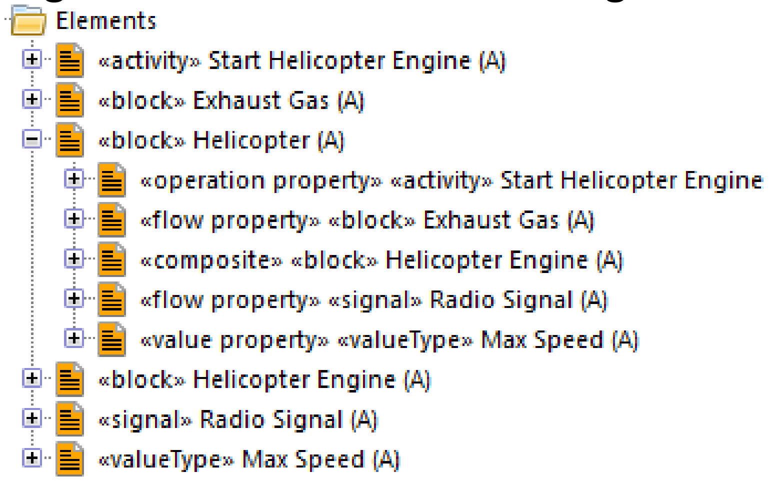
By themselves these elements have limited value, but once they are linked together using **relationships** the connected elements tell a story. See next slide.

MBSE - Information Modeling Language :2

In the following example there are four different kinds of **relationships** used to connect the different types of elements to tell a story. The example below tells the story that a helicopter has an Engine, it performs a Start Engine activity, it has a Max Speed variable, it inputs Radio Signals and outputs Exhaust Gas.



An alternative way to view an element’s related information is in a browser tree view as shown below. Elements can be created and linked to other elements using relationships in the browser without having to create a box and line diagram.

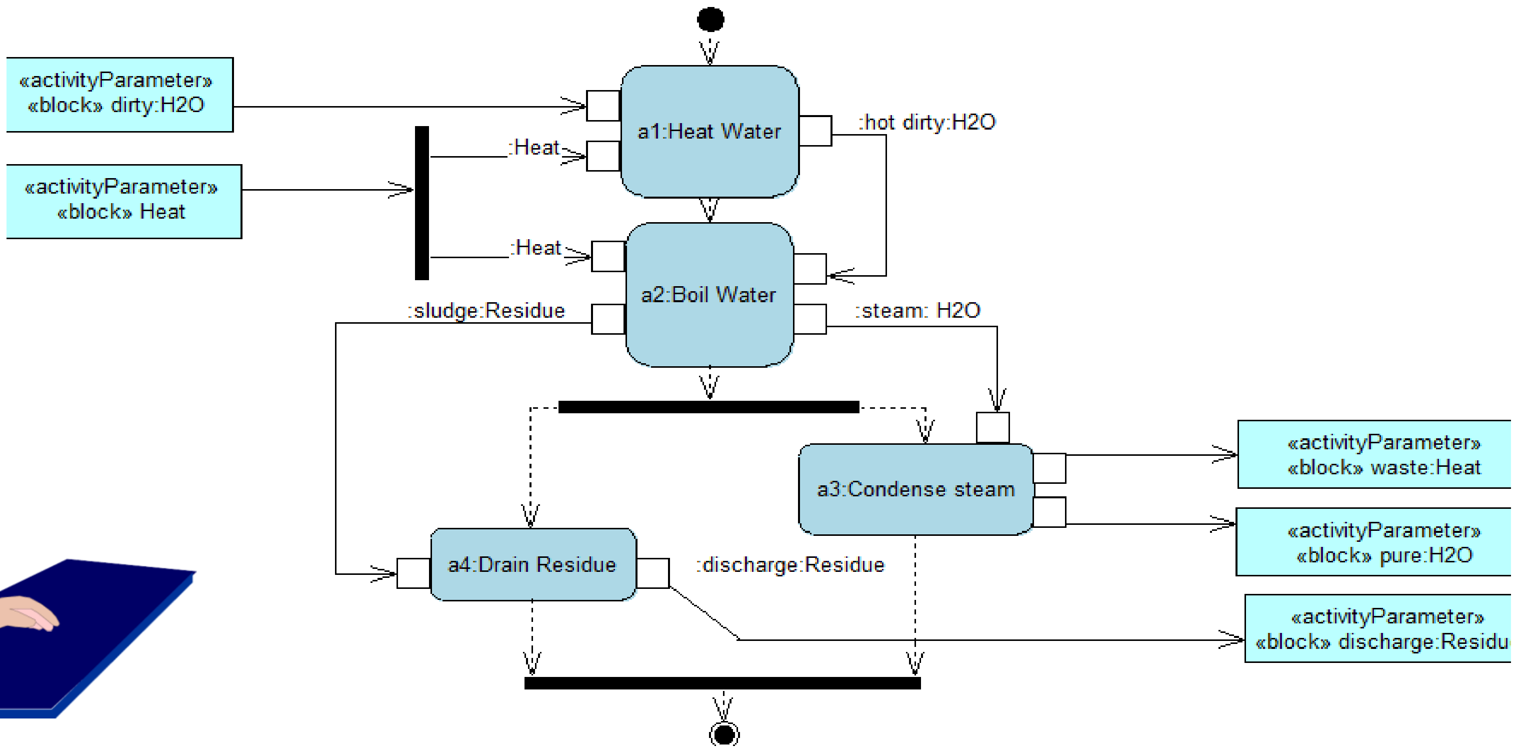


Elements
- «activity» Start Helicopter Engine (A)
- «block» Exhaust Gas (A)
- «block» Helicopter (A)
  - «operation property» «activity» Start Helicopter Engine
  - «flow property» «block» Exhaust Gas (A)
  - «composite» «block» Helicopter Engine (A)
  - «flow property» «signal» Radio Signal (A)
  - «value property» «valueType» Max Speed (A)
- «block» Helicopter Engine (A)
- «signal» Radio Signal (A)
- «valueType» Max Speed (A)

MBSE - Information Modeling Language :3

Some Concepts / Stories are best communicated with Diagrams (i.e., conceptual model)

- System engineers create diagrams to better understand problems and develop candidate solutions, and validate design decisions.
- Diagrams <u>improve communication</u> between team members, and with the customer.

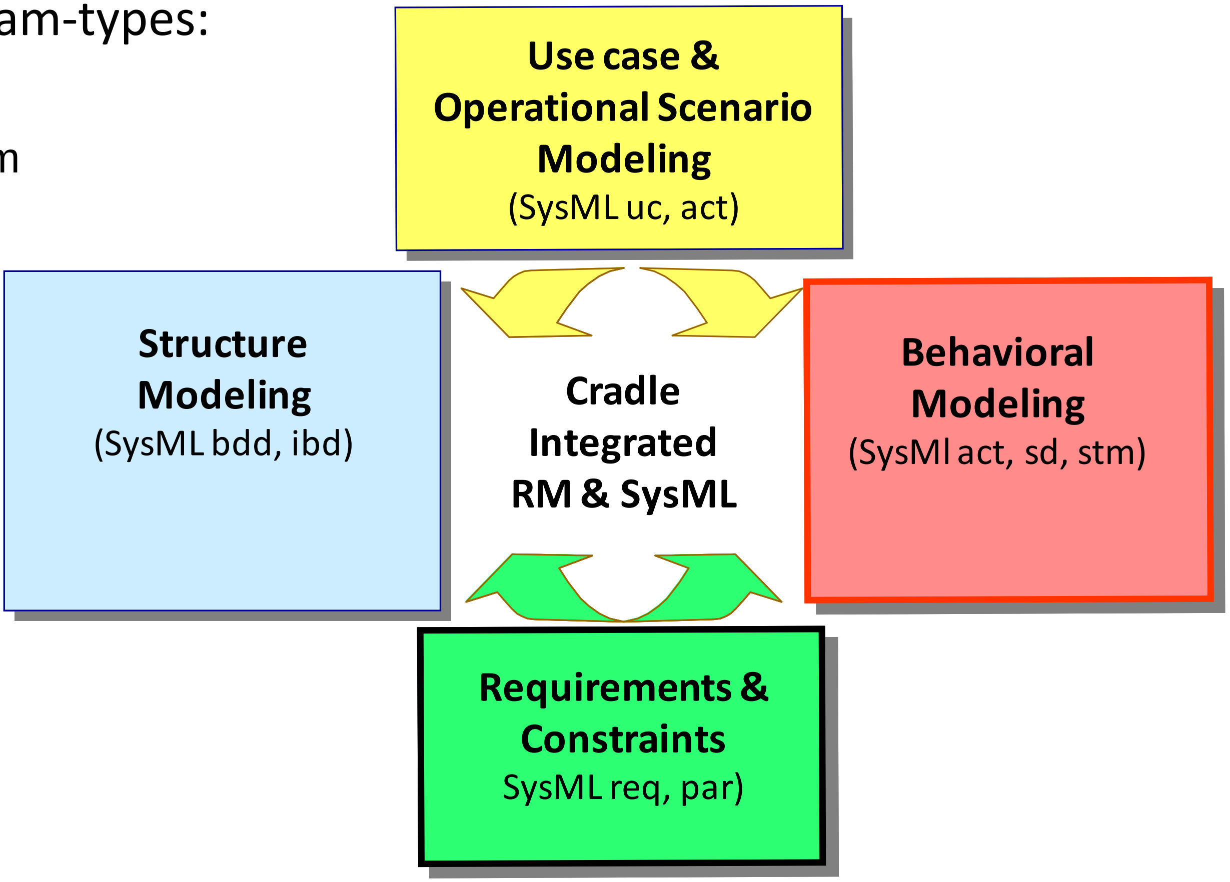


«activityParameter» «block» dirty:H2O
«activityParameter» «block» Heat
:Heat
:Heat
a1:Heat Water
:hot dirty:H2O
a2:Boil Water
:sludge:Residue
:steam: H2O
a3:Condense steam
a4:Drain Residue
:discharge:Residue
«activityParameter» «block» waste:Heat
«activityParameter» «block» pure:H2O
«activityParameter» «block» discharge:Residu

MBSE - Information Modeling Language :4

SysML specifies Nine diagram-types:

- uc - Use Case Diagram
- bdd - Block Definition Diagram
- ibd - Internal Block Diagram
- act - Activity Diagram
- sd - Sequence Diagram
- stm - State Machine Diagram
- req - Requirements Diagram
- par - Parametric Diagram
- pkg - Package Diagram

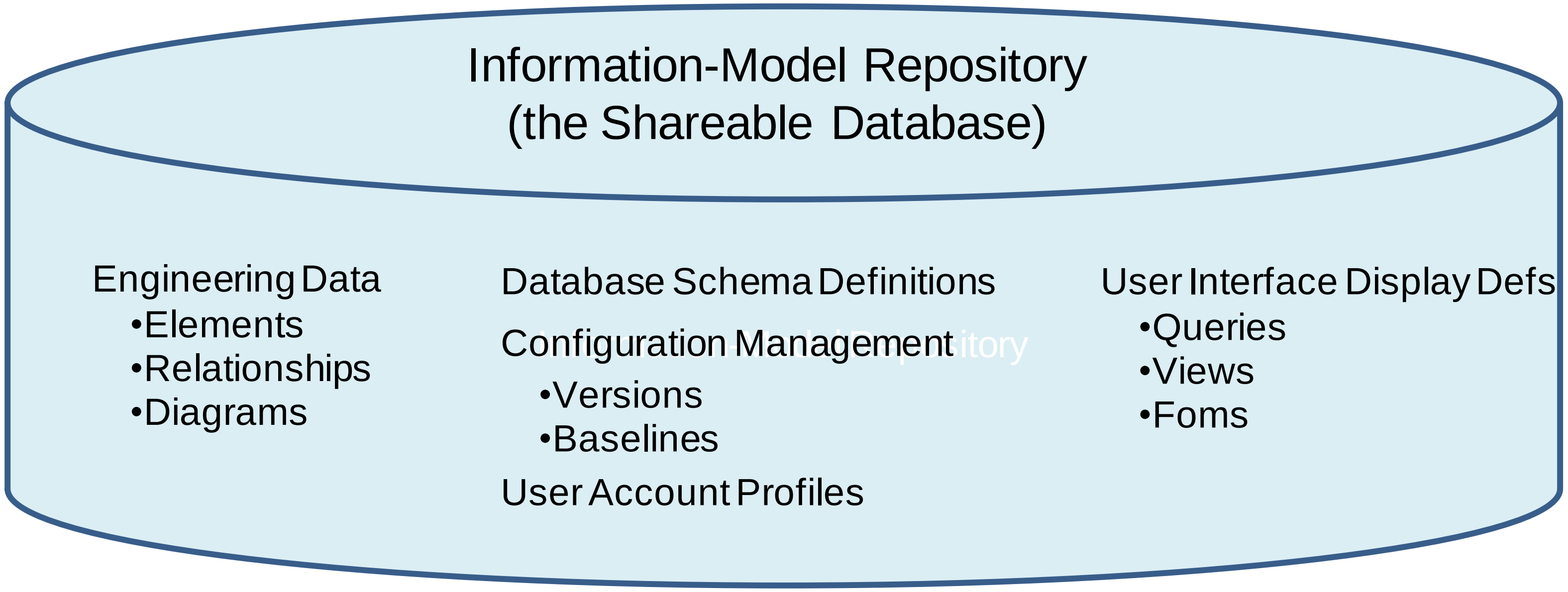

Use case & Operational Scenario Modeling (SysML uc, act)
Structure Modeling (SysML bdd, ibd)
Cradle Integrated RM & SysML
Behavioral Modeling (SysMl act, sd, stm)
Requirements & Constraints SysML req, par)

Many organizations are adopting these nine diagram-types so there will be common graphical notations for specifying candidate design solutions in order to improve project personnel’s ability to communicate.

MBSE – Information-Model Repository

An Information-Model Repository accessible by all project personnel is required to share the model information (i.e., stereotypes elements, relationships, and diagrams).

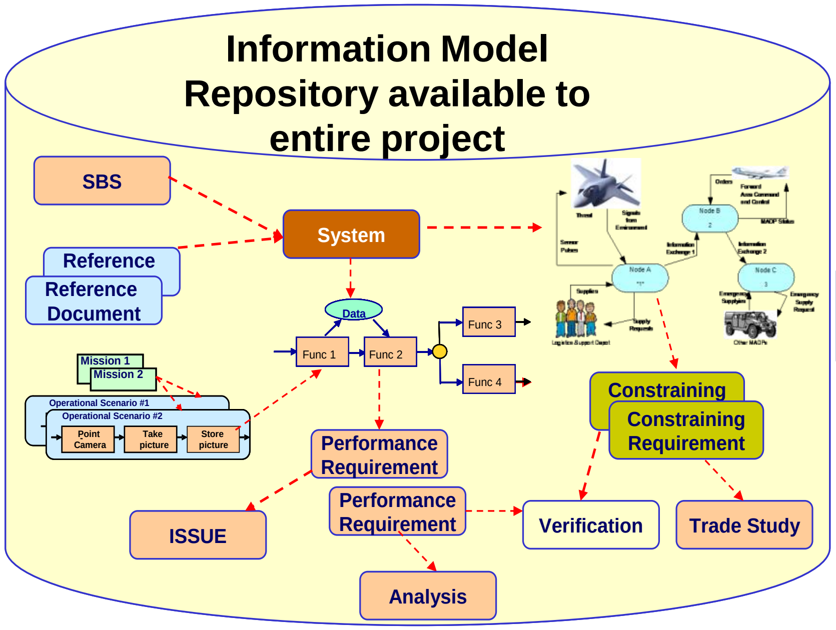
The MBSE Information Repository stores and manages multiple Information models and variants of the models. A model may consist of unique project information as well as common information shareable by any project. The Repository holds all the meta-data that makes up the Information Model.



MBSE – Automated Tools

Automated Tools are needed to load information into the repository and extract information for publishing have proven there worth to projects that utilized the MBSE process.

Tools to Capture and edit Information in the Repository



Tools to Publish Model Information:

•Requirements and Architecture Documents

•Interface Specifications

•Metrics & Traceability

•Baselines

•Analysis Reports

•Review Packages

• XML/XMI data exchanges with detailed design tools

•Automatic Document Generation

3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

SysML Stereotype Elements Defined (based on SysML Spec 1.3)

The language can be extended with new elements and new relationships

SysML Stereotype Elements

The definition of the SysML Modeling Elements are as follows:

SysML Stereotype	Description	Diagram Type
«acceptChangeAction»	An accept change action responds to a conditional event such as change in a «valueType» or resource «block» element.	act
«acceptSignalAction»	An accept signal action responds to the receipt of a «signal» element..	act
«action»	Actions execute and transform their inputs into outputs. Also known as functions.	act
«activityParameterSet»	The input/ouputt activity parameters of an activity can be grouped together into parameter sets.	act
«activityParameter»	An input/output activity parameter models the flow of matter, energy, or data into and/or out of an activity diagram. An activity parameter can be represented by the following: «block» or «valueType».	act
«activity»	An activity specifies the transformation of inputs to outputs. Associate an activity with an activity diagram to specify a controlled sequence of action executions..	bdd
«actor»	An actor represents the role of a human, an organization, or any external system that interacts with the system being developed.	uc, bdd, sd
«block»	A block is a unit of structure used to define a type of thing (e.g., system or item that flows through the system).	uc, bdd, ibd, sd

SysML Stereotype Elements :2

«callBehaviourAction»	A call behaviour action can invoke, i.e., call) another behaviour diagram (act, stm, sd) for execution. In other words one behaviour diagram can call another behaviour diagram.	act
«callOperationAction»	A call operation action can invoke/call an operation (i.e., activity) linked to a specific block with the operation property relationship.	act
«centralBuffer»	A central buffer provides storage for multiple data objects of the same type before the objects are consumed by downstream actions.	act
«constraint»	A constraint is used to define performance constraints on the value properties of blocks and activities.	bdd, par
«controlFlowEdge»	Control flow edge (dashed line with open arrowhead) establish the order in which the actions within an activity diagram will execute.	act
«controlOperatorAction»	This action invokes/calls a activity diagram, associated with **«**controlOperator**t»**, to define control parameters for enabling and disabling actions on the calling diagram.	act
«controlOperator»	A controlOperator is associated with an activity diagram to define control parameters for enabling and disabling other actions.	act
«dataStore»	A dataStore provides storage for a data object before the object is consumed by a downstream action.	act

SysML Stereotype Elements :3

«entryPoint»	An entryPoint can be used to define different entry points into a composite state or another state machine.	stm
«enumeration»	An enumeration defines a set of named values called literals that can be referenced by a **«**valueType**»**.	bdd
«exitPoint»	An exitPoint can be used to define different exit points out of a composite state or another state machine.	stm
«fragment»	An interaction/sequence diagram can model a group of event occurrences using a special construct called a combined fragment.	sd
«fullPort»	A fullPort assigned to a block has a port definition block that defines valid incoming and/or outgoing flow items.	bdd, ibd
«historyPseudoState»	A history pseudo-state identifies the last active state when a composite state was last exited.	stm
«interaction»	An interaction is a unit of behaviour that focuses on the observable exchange of messages between blocks and/or actors using sequence diagram. The «interaction» is linked to a specific sequence diagram.	sd
«interfaceBlock»	An interfaceBlock is a kind of entity that defines a set of public features (i.e., operations and flow properties) a proxy port must support. Therefore, a proxy port is linked to an interface block.	bdd. ibd

SysML Stereotype Elements :4

«interruptingEdge»	An interrupting edge interrupts the execution of the actions in an interruptible region of activity diagram.	act
«libraryModel»	A libraryModel package that contains model elements that are intended to be reused by other packages.	pkg
«lifeLine»	A lifeline represents an individual participant (block or actor) in an Interaction / Sequence diagram.	sd
«message»	A Message defines a particular communication between Lifelines of an Interaction/Sequence Diagram.	sd
«model»	A model package is a special type of package that groups regular packages into a hierarchy of packages.	pkg
«objectFlowEdge»	An object flow is an activity diagram edge (solid line with open arrowhead) that can have objects or data passing along it.	act
«objectNode»	An object node is an entity that flows through an activity/activity diagram. An objectNode symbol can represent the following types of elements: «block», «signal», or «valueType».	act
«package»	A package is a container/namespace for model elements, diagrams, and nested packages.	pkg

SysML Stereotype Elements :5

«pin»	A pin is an object node identifying the inputs and outputs to/from actions on activity diagrams. An pin symbol can represent the following types of elements: «block», «signal», or «valueType».	act
«proxyPort»	A proxyPort assigned to a block has a port definition interfaceBlock that defines valid incoming and/or outgoing items. A proxy port is linked to an interface block.	bdd, ibd
«sendSignalAction»	The sendSignal action passes a signal to a receiver entity.	act, stm
«signal»	A signal defines an asynchronous event that can be sent and received by a block. A signal is defined on a bdd or ibd, and used on the following: act, sd, stm.	bdd, ibd, act, sd, stm
«stateMachine»	A stateMachine specifies the sequences of states that a block goes through during its life in response to events, together with its responses and actions. The «stateMachine» is linked to a specific State Machine diagram.	stm
«state»	A state represents some significant condition in the life of a block.	stm
«submachineState»	A state machine may be reused using a kind of state called submachine state that invokes the reused state machine.	stm
«transition»	A transition specifies when a change of state occurs within a state machine. A transition may have a 'trigger event', a 'guard', and a 'behaviour effect' associated with the transition. A trigger event can be either a call event, signal event, or a change event.	stm
«useCase»	A use case describes how users/stakeholders use the proposed system to achieve their goals.	uc

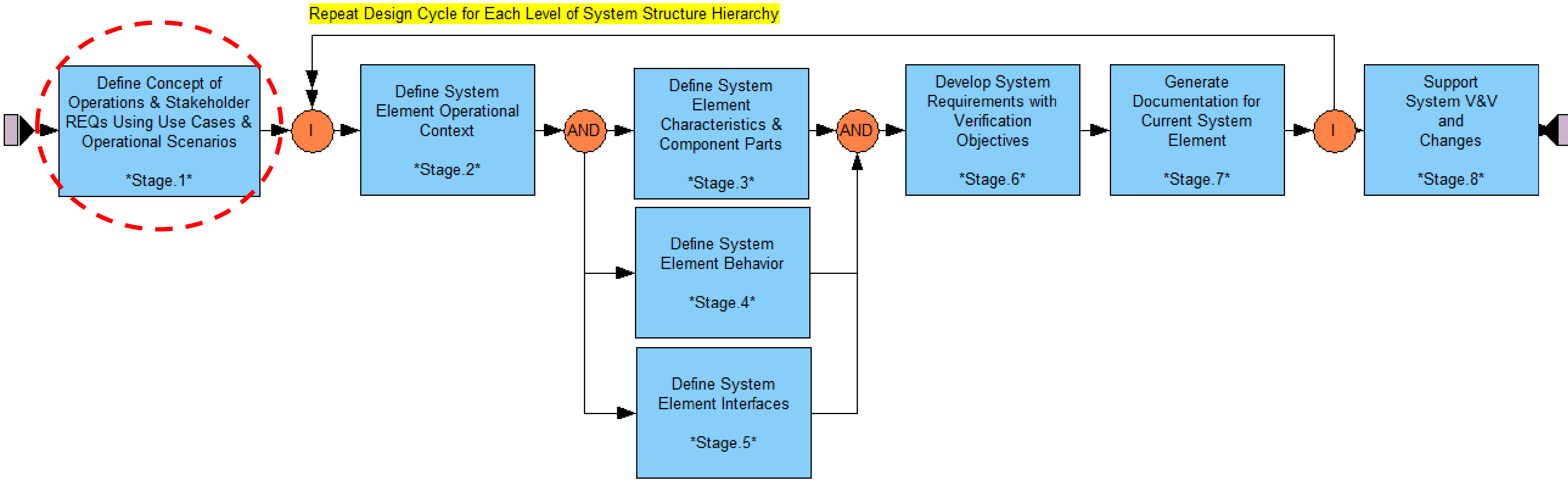
SysML Stereotype Elements :6

«valueType»	A valueType is a form of data value (e.g. integer, real, boolean, string) with a specified unit of measurement, a defined kind of quantity, and a value. It is used to define block value properties, as arguments for signals, as I/O parameters for behaviour elements. This type of element is defined on the bdd and ibd and used on all the other diagram types except for req and uc.	bdd, ibd, act, sd, stm, pkg, par
«viewPoint»	A viewpoint is used to specify conventions and rules for constructing and using a view for the purpose of addressing a set of stakeholder concerns.	pkg
«view»	A view is a special type of package that lists the model elements that support a particular stakeholder perspective.	pkg
«waitTimeAction»	A waitTime action corresponds to an expiration of an implicit timer in an activity diagram. A relative time is shown as ‘after(xxx)’, while an absolue time is shown as ‘at(xxx)’	act

Overview of a MBSE Process Using SysML

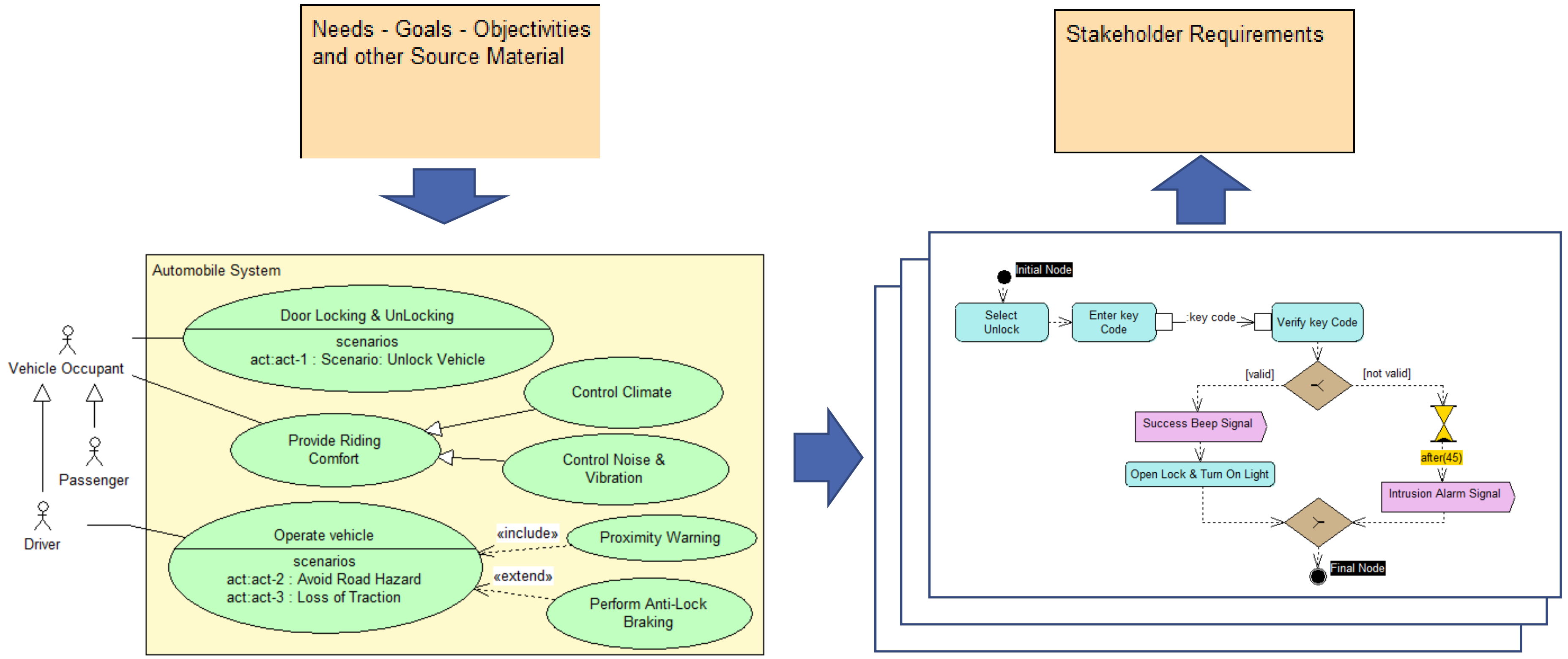
This process was used successfully on a NASA project using the Cradle Systems Engineering tool

Stage 1- Define Concept of Operations Using Use Cases & Operational Scenarios & Define Stakeholder REQs


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

Stage 1- Define Concept of Operations Using Use Cases & Operational Scenarios & Define Stakeholder REQs

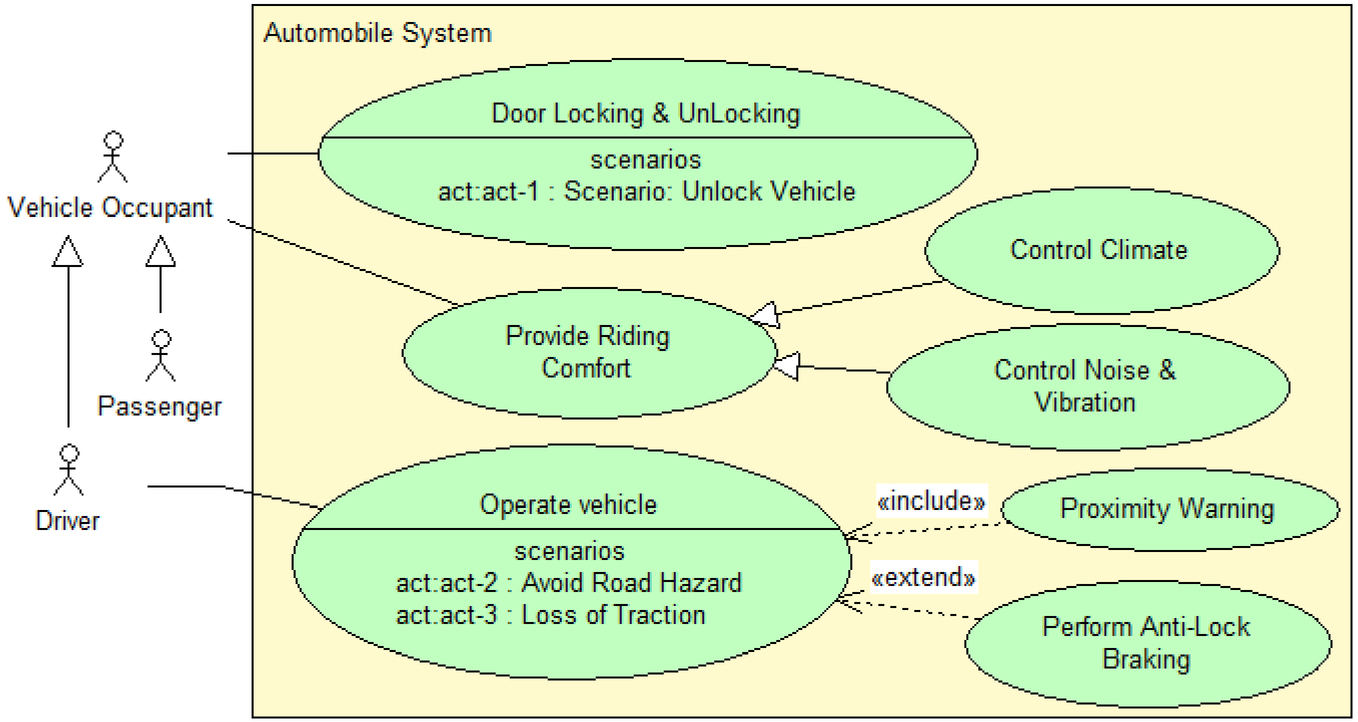
The tasks in this stage are performed at project start-up to define the project scope, collect Stakeholder Needs and Goals, prepare a concept of operations (ConOps), and then develop the set of Stakeholder / User Requirements and publish Stakeholder Requirements Document.



Use Cases **Operational Scenarios**

SysML Use Case Diagram (uc)

A Use Case Diagram displays a set of use cases (the missions/goals/features/capabilities for the subject system) as well as the actors/stakeholders that invoke and participate in those use cases. Should be from users point of view not system developer. A Use Case is described by a set of operational scenarios (see next slide).



- Stick Figure: Actor / Stakeholder
- Hollow Triangle: specialized configuration/variant on other end of path
- Rectangle: Subject System under development

- Oval: Use case that specifies a mission or goal.
- Solid Line with out adornments: Association between actor and Use Case
- Dashed Extend Arrow: Extends functionality of a use case

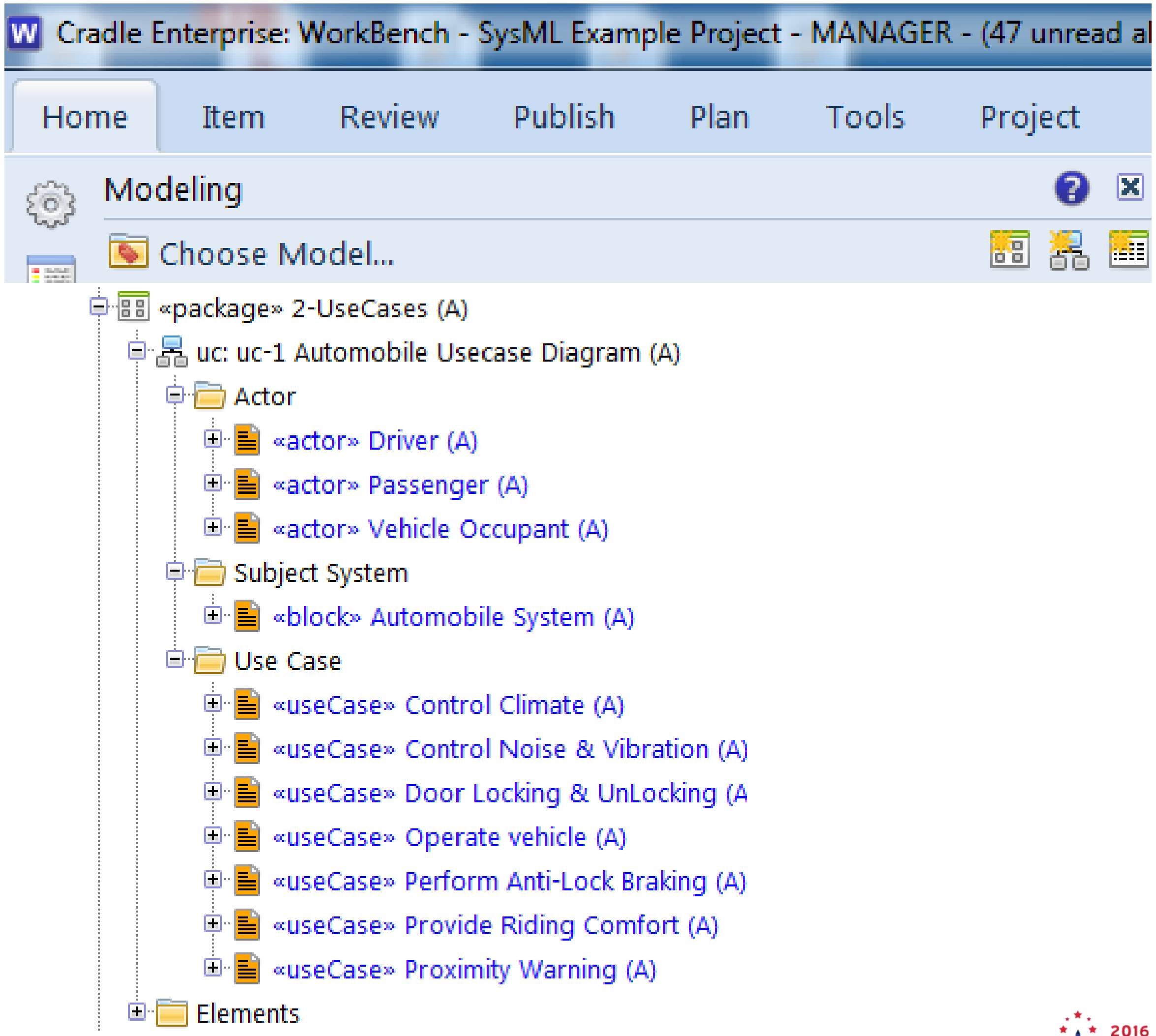
- Dashed Include Arrow: Includes functionality of another use case
- Oval Scenario Compartment: Operational scenarios that describe usecase

SysML Stereotypes Used on Sample uc Diagram

«actor»

«block»

«useCase»

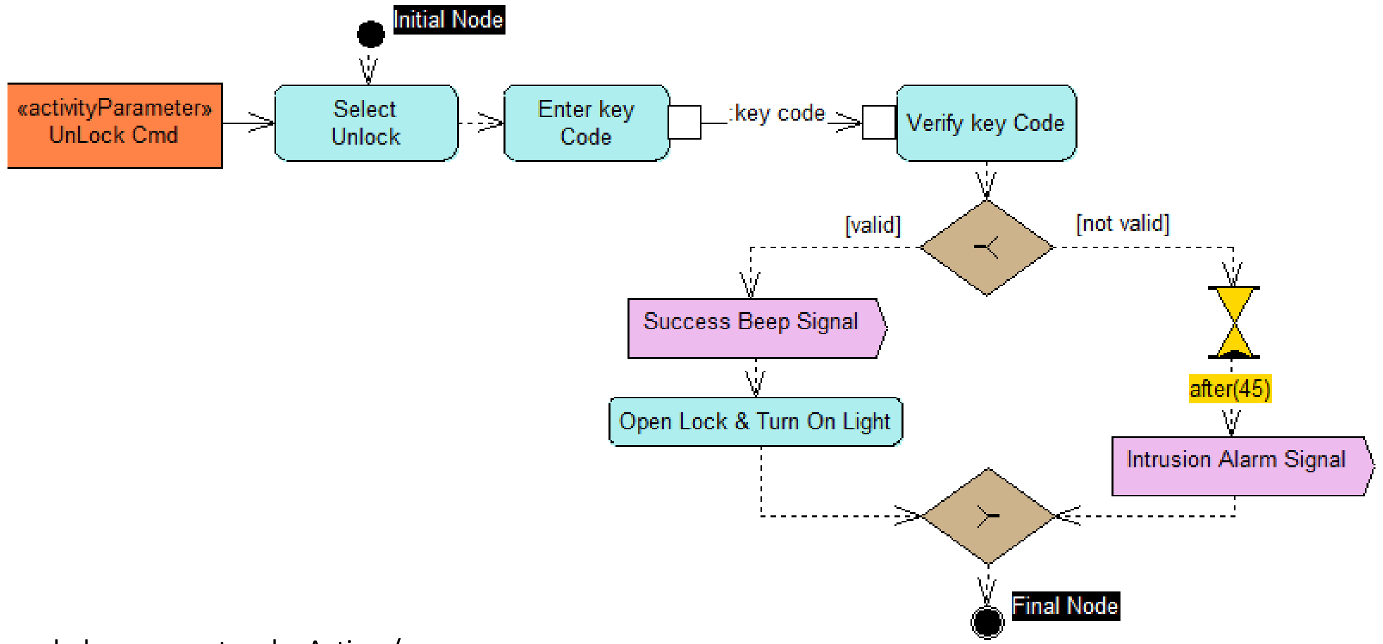


Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

- «package» 2-UseCases (A)
  - uc: uc-1 Automobile Usecase Diagram (A)
    - Actor
      - «actor» Driver (A)
      - «actor» Passenger (A)
      - «actor» Vehicle Occupant (A)
    - Subject System
      - «block» Automobile System (A)
    - Use Case
      - «useCase» Control Climate (A)
      - «useCase» Control Noise & Vibration (A)
      - «useCase» Door Locking & UnLocking (A
      - «useCase» Operate vehicle (A)
      - «useCase» Perform Anti-Lock Braking (A)
      - «useCase» Provide Riding Comfort (A)
      - «useCase» Proximity Warning (A)
  - Elements

Operational Scenario Using SysML Activity Diagram (act)

To understand a use case we tell stories. These stories cover how to successfully achieve the goal/mission/feature/capability, and how to handle any faults/problems that may occur. The details of the stories are captured in SysML Activity Diagrams. These diagrams used in this context are referred to as operational scenarios as primary component of the concept of operations document.



- Rounded corner rectangle: Action / function
- Rectangle: Activity Parameter I/O
- Small Square on outside edge of Action: Pin representing flow object

- Solid black bar: Fork indicates parallel processing
- Dashed arrow: Control Flow
- Solid arrow: Object Flow
- Rectangle with triangle on left side: Accept Signal/Change Action:
- Diamond: Decision Node & merge Node

- Bulls-eye circle: Active Final Node
- Swim lanes: Performer (System Element)
- Solid black circle: Initial Node of the activity

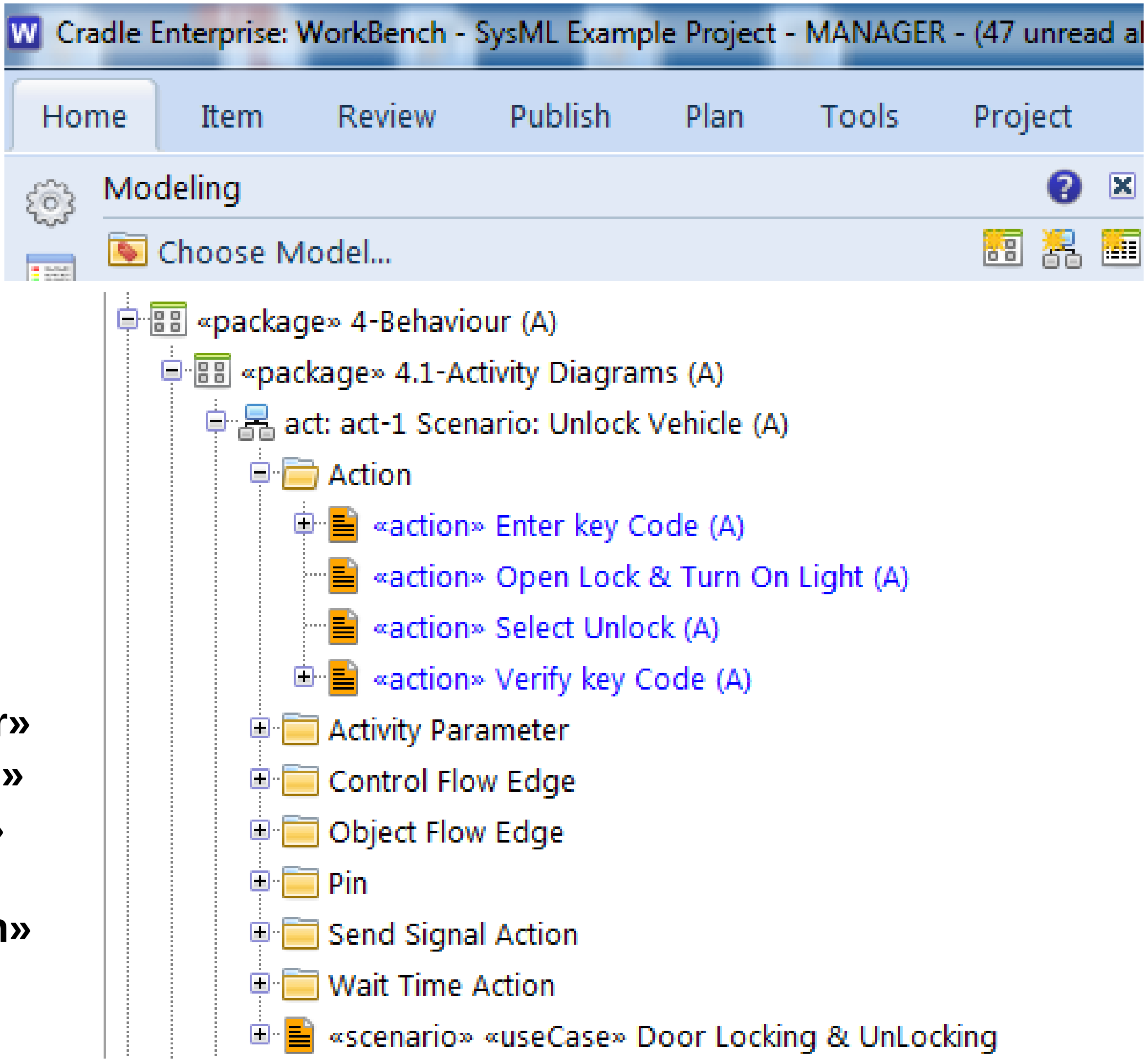
3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

SysML Stereotypes Used on Sample act Diagram

«action»

«activityParameter»
«controlFlowEdge»
«objectFlowEdge»
«pin»
«sendSignalAction»
«waitTimeAction»



Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

Home | Item | Review | Publish | Plan | Tools | Project

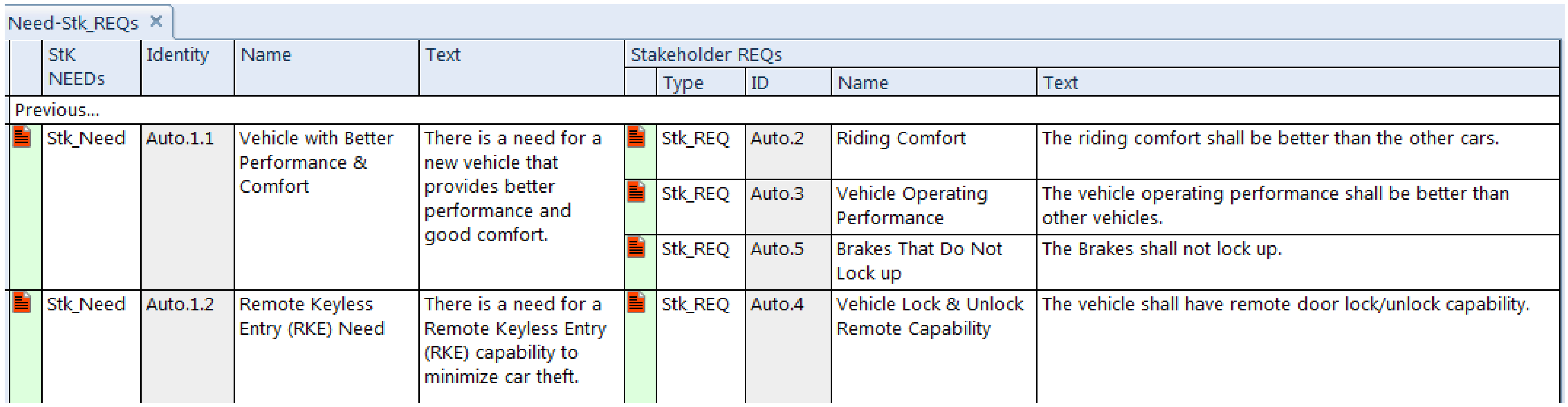
Modeling

Choose Model...

- «package» 4-Behaviour (A)
  - «package» 4.1-Activity Diagrams (A)
    - act: act-1 Scenario: Unlock Vehicle (A)
      - Action
        - «action» Enter key Code (A)
        - «action» Open Lock & Turn On Light (A)
        - «action» Select Unlock (A)
        - «action» Verify key Code (A)
      - Activity Parameter
      - Control Flow Edge
      - Object Flow Edge
      - Pin
      - Send Signal Action
      - Wait Time Action
      - «scenario» «useCase» Door Locking & UnLocking

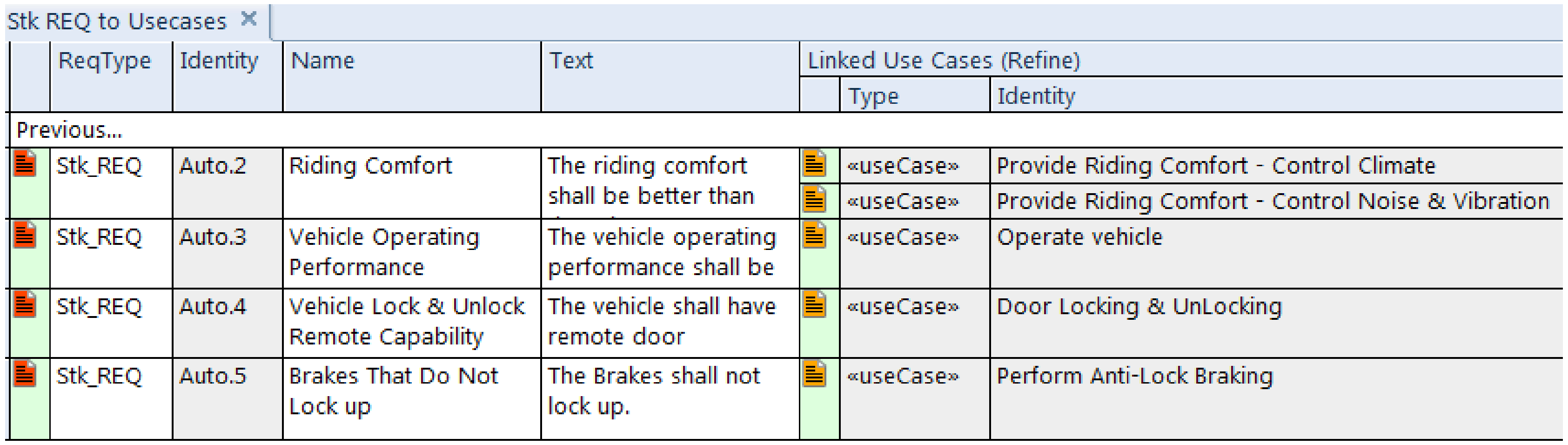
Stakeholder Needs and Stakeholder Requirements

The following example shows two requirement traceability tables. The first contains traceability between the Stakeholder Needs and the derived Stakeholder Requirements. The second table shows derived Stakeholder REQs (black box requirements) with traceability to the Use Cases.



Need-Stk_REQs

StK NEEDs	Identity	Name	Text	Stakeholder REQs			
				Type	ID	Name	Text
Previous...							
Stk_Need	Auto.1.1	Vehicle with Better Performance & Comfort	There is a need for a new vehicle that provides better performance and good comfort.	Stk_REQ	Auto.2	Riding Comfort	The riding comfort shall be better than the other cars.
				Stk_REQ	Auto.3	Vehicle Operating Performance	The vehicle operating performance shall be better than other vehicles.
				Stk_REQ	Auto.5	Brakes That Do Not Lock up	The Brakes shall not lock up.
Stk_Need	Auto.1.2	Remote Keyless Entry (RKE) Need	There is a need for a Remote Keyless Entry (RKE) capability to minimize car theft.	Stk_REQ	Auto.4	Vehicle Lock & Unlock Remote Capability	The vehicle shall have remote door lock/unlock capability.

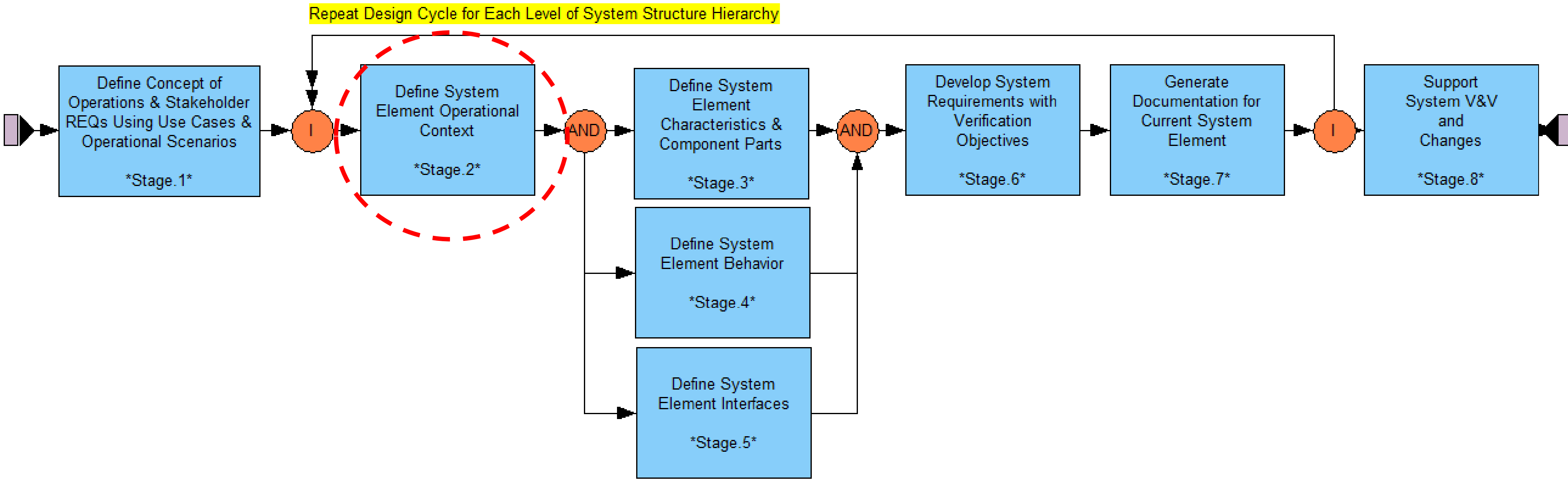


Stk REQ to Usecases

ReqType	Identity	Name	Text	Linked Use Cases (Refine)	
				Type	Identity
Previous...					
Stk_REQ	Auto.2	Riding Comfort	The riding comfort shall be better than	«useCase»	Provide Riding Comfort - Control Climate
				«useCase»	Provide Riding Comfort - Control Noise & Vibration
Stk_REQ	Auto.3	Vehicle Operating Performance	The vehicle operating performance shall be	«useCase»	Operate vehicle
Stk_REQ	Auto.4	Vehicle Lock & Unlock Remote Capability	The vehicle shall have remote door	«useCase»	Door Locking & UnLocking
Stk_REQ	Auto.5	Brakes That Do Not Lock up	The Brakes shall not lock up.	«useCase»	Perform Anti-Lock Braking

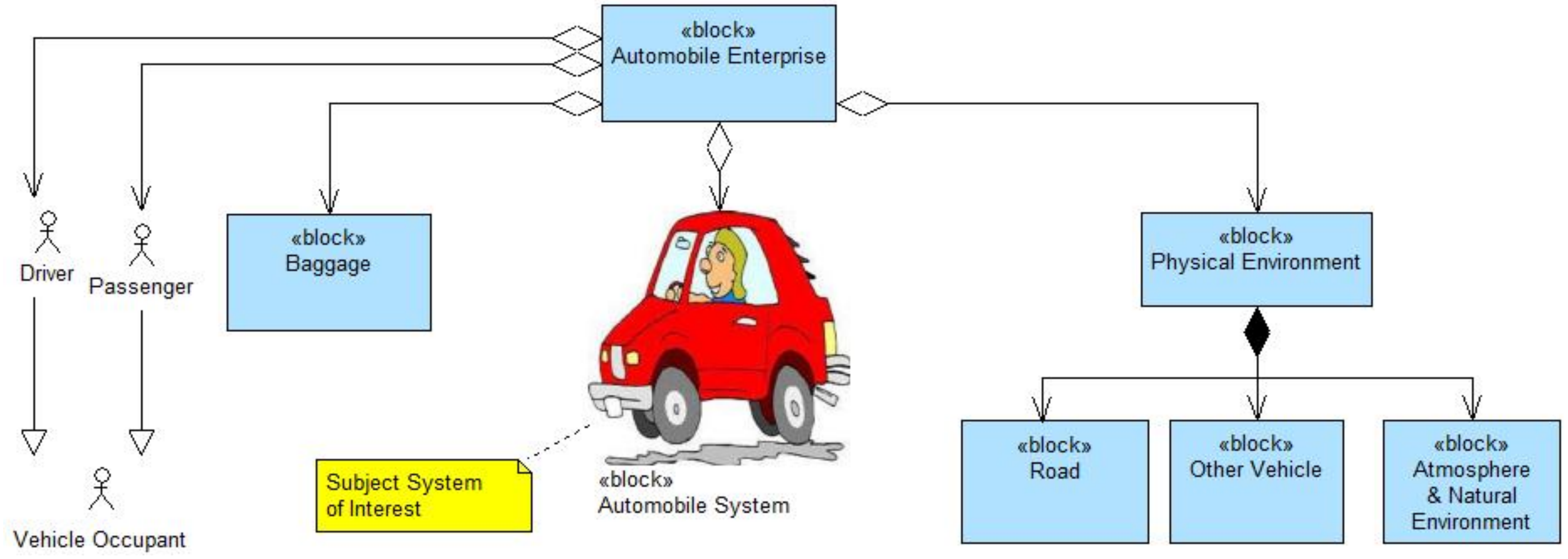
- StK_Need: Stakeholder need statement
- Stk_REQ: Stakeholder requirement statement

Stage 2 - Define System Element Operational Context


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

Stage 2 - Define System Element Operational Context

An Operational Context Diagram is created for the Current System Element in the System Structure Hierarchy. The purpose is to identify all external systems (natural environment and manmade) that can interact with the system element and identify the top-level external interfaces. A Block Definition Diagram (bdd) is used to show external systems. An Internal Block Diagram (ibd) is used to show the external interfacing information. The following is a bdd.

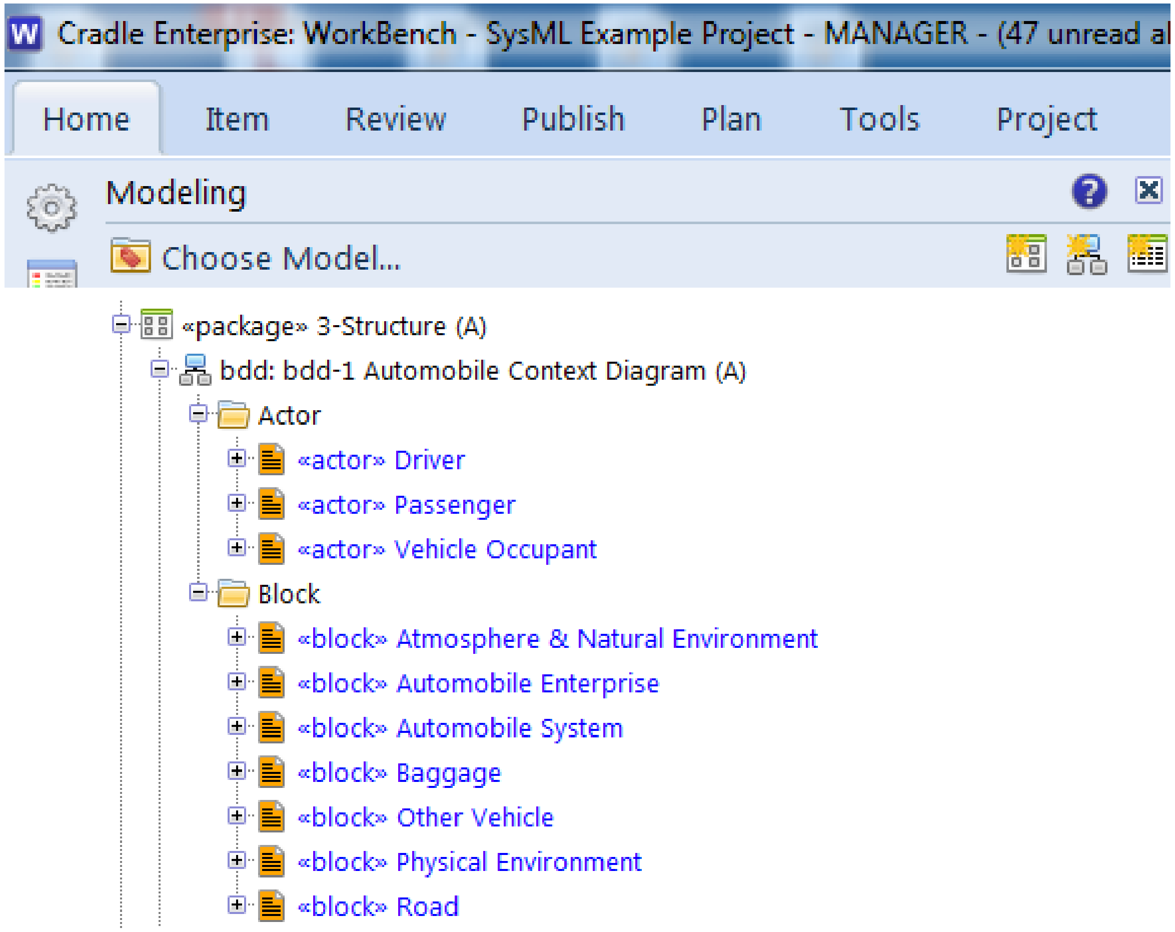


- Rectangle: A «block» element is a type of thing (system, subsystem, organization, report, etc)
- White Diamond: Reference / Shared Blocks on other end of path
- Solid Diamond: Composed of Part Blocks on other end of path
- Hollow Triangle: specialized configuration/variant on other end of path

SysML Stereotypes Used on Sample bdd Diagram

«actor»

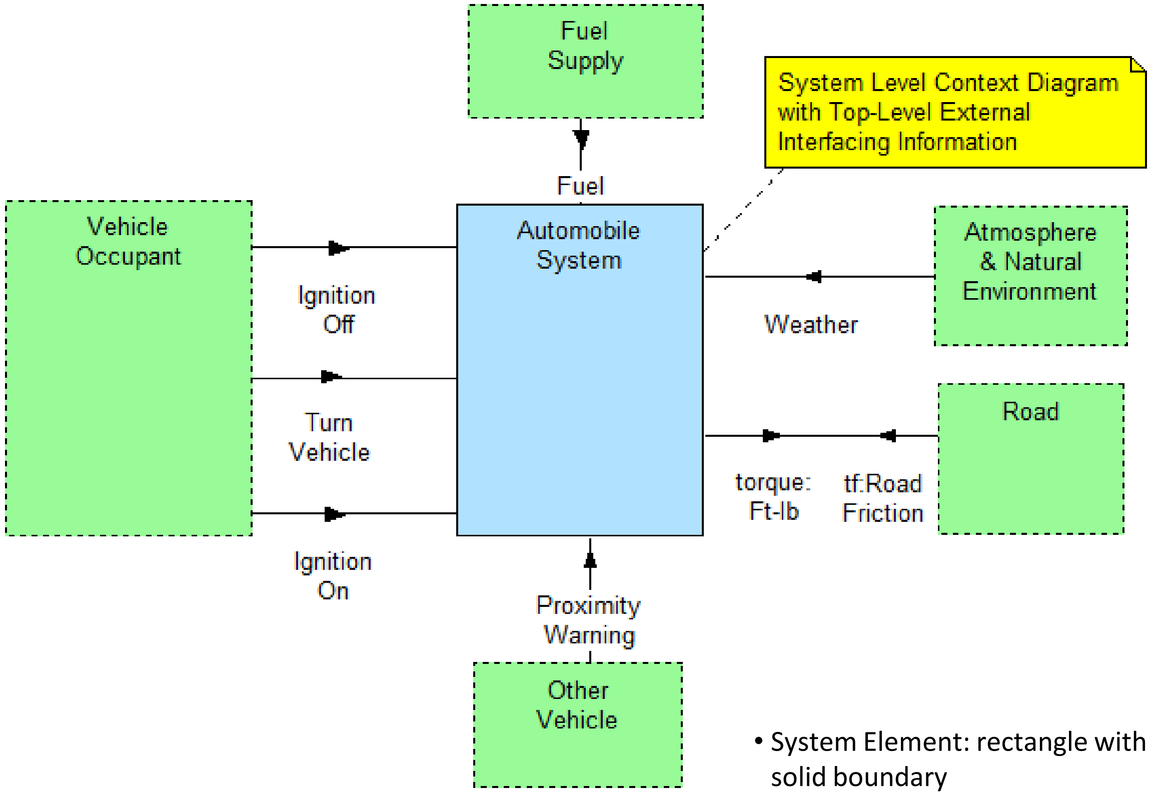
«block»


Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

- «package» 3-Structure (A)
  - bdd: bdd-1 Automobile Context Diagram (A)
    - Actor
      - «actor» Driver
      - «actor» Passenger
      - «actor» Vehicle Occupant
    - Block
      - «block» Atmosphere & Natural Environment
      - «block» Automobile Enterprise
      - «block» Automobile System
      - «block» Baggage
      - «block» Other Vehicle
      - «block» Physical Environment
      - «block» Road

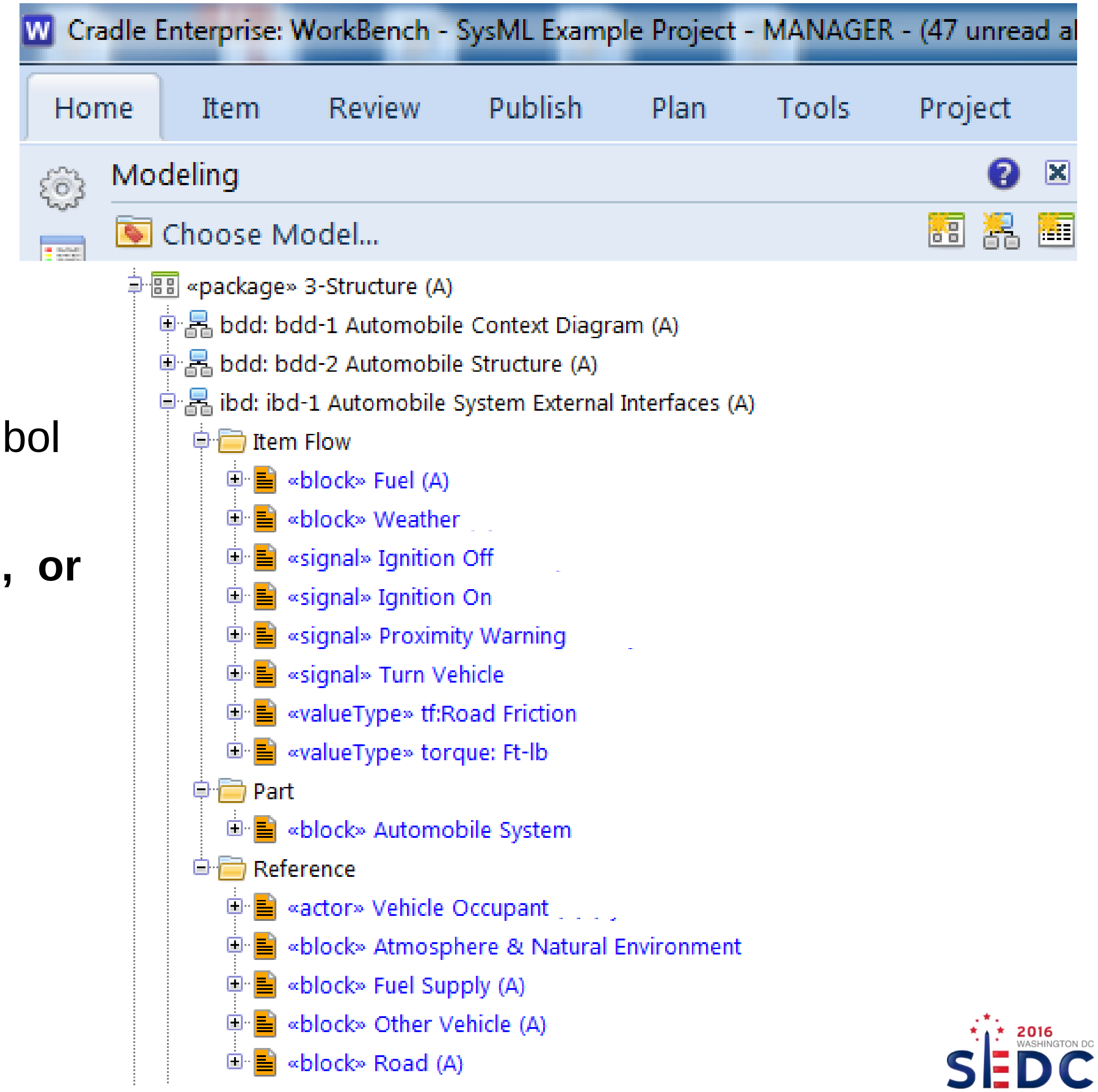
Stage 2 - Define System Element Operational Context :2

This ibd shows **external systems** as dashed rectangle symbols and the **interfacing information** with the annotated arrowheads on the connector lines. Why is this diagram important? If we miss an external system or an interface we may fail to identify a needed requirement making our design specification incomplete.



- System Element: rectangle with solid boundary
- External System: rectangle with dashed boundary
- interfacing info (Item Flow): solid arrowhead straddling connectors

SysML Stereotypes Used on Sample ibd Diagram



Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

Home | Item | Review | Publish | Plan | Tools | Project

Modeling

Choose Model...

- «package» 3-Structure (A)
  - bdd: bdd-1 Automobile Context Diagram (A)
  - bdd: bdd-2 Automobile Structure (A)
  - ibd: ibd-1 Automobile System External Interfaces (A)
    - Item Flow
      - «block» Fuel (A)
      - «block» Weather
      - «signal» Ignition Off
      - «signal» Ignition On
      - «signal» Proximity Warning
      - «signal» Turn Vehicle
      - «valueType» tf:Road Friction
      - «valueType» torque: Ft-lb
    - Part
      - «block» Automobile System
    - Reference
      - «actor» Vehicle Occupant
      - «block» Atmosphere & Natural Environment
      - «block» Fuel Supply (A)
      - «block» Other Vehicle (A)
      - «block» Road (A)

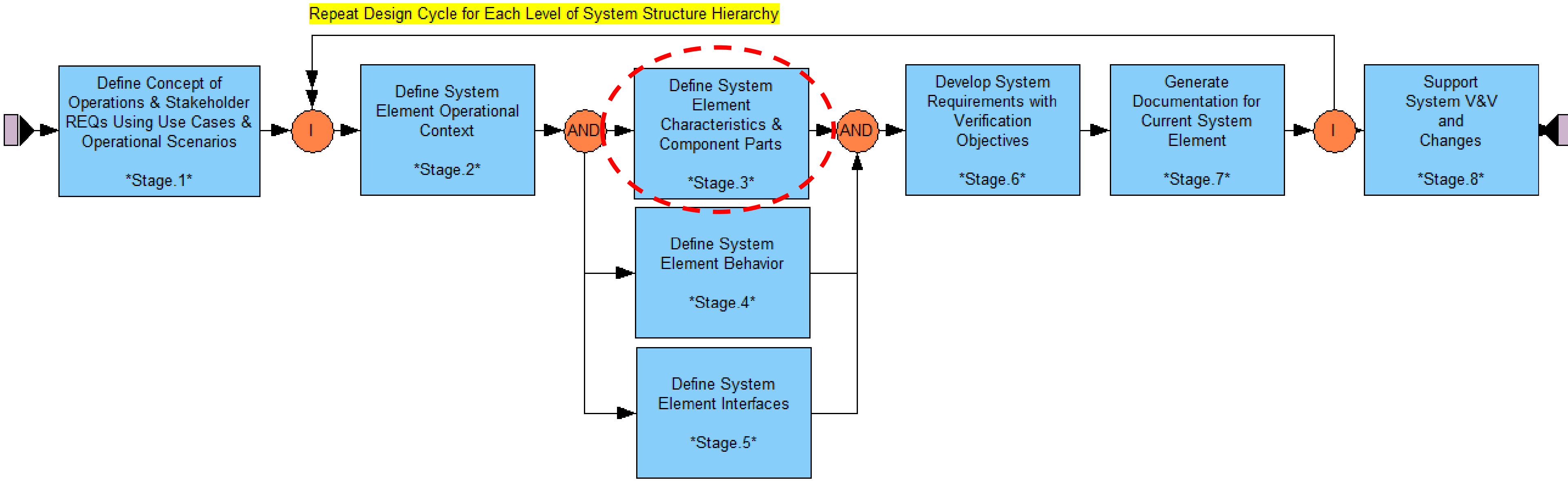
Item Flow is a symbol Represented by:

«block», «signal», or «valueType»

«block»

«block», «actor»

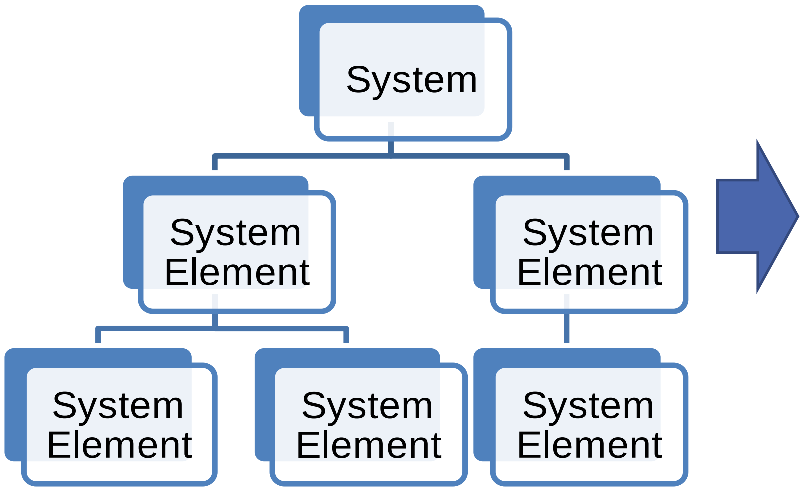
Stage 3 - Define System Element Characteristics & Component Parts


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

Stage 3 - Define System Element Characteristics & Component Parts

Based on information captured in the previous stages of the systems engineering process, and on-going trade studies, define the current system element's physical characteristics and its component parts one level down the system structure hierarchy.

Why is it important to capture the system structure in the Information-Model? Because the systems engineers must identify behavior, external I/O, component parts, performance constraints, and identify requirements for each block in the hierarchy. Once the behavior for a specific block is defined, the engineers can then identify the best component parts to use to accomplish that candidate behavior. Knowing the allocated behavior, the behavioral requirements can be defined.



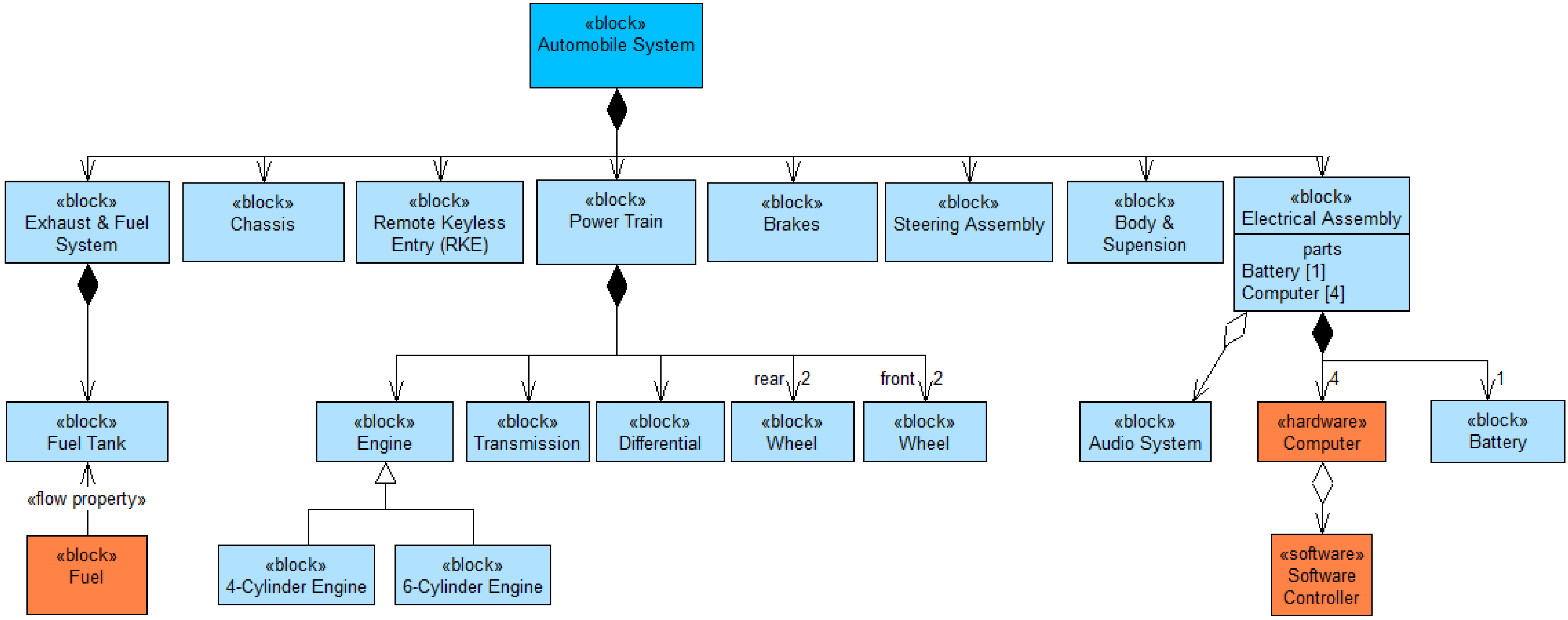
System / System Element	Specification Information to be captured:
Physical Characteristics	Stage 3
Component Parts	Stage 3
Element's Behavior	Stage 4
Element's Interfaces	Stage 5
Element's Constraints	Stage 3, 4, 5
Element's Requirements	Stage 3, 4, 5

3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

System Structure Hierarchy Using SysML bdd

In the following SysML bdd the systems engineer is showing multiple levels in the structure hierarchy.

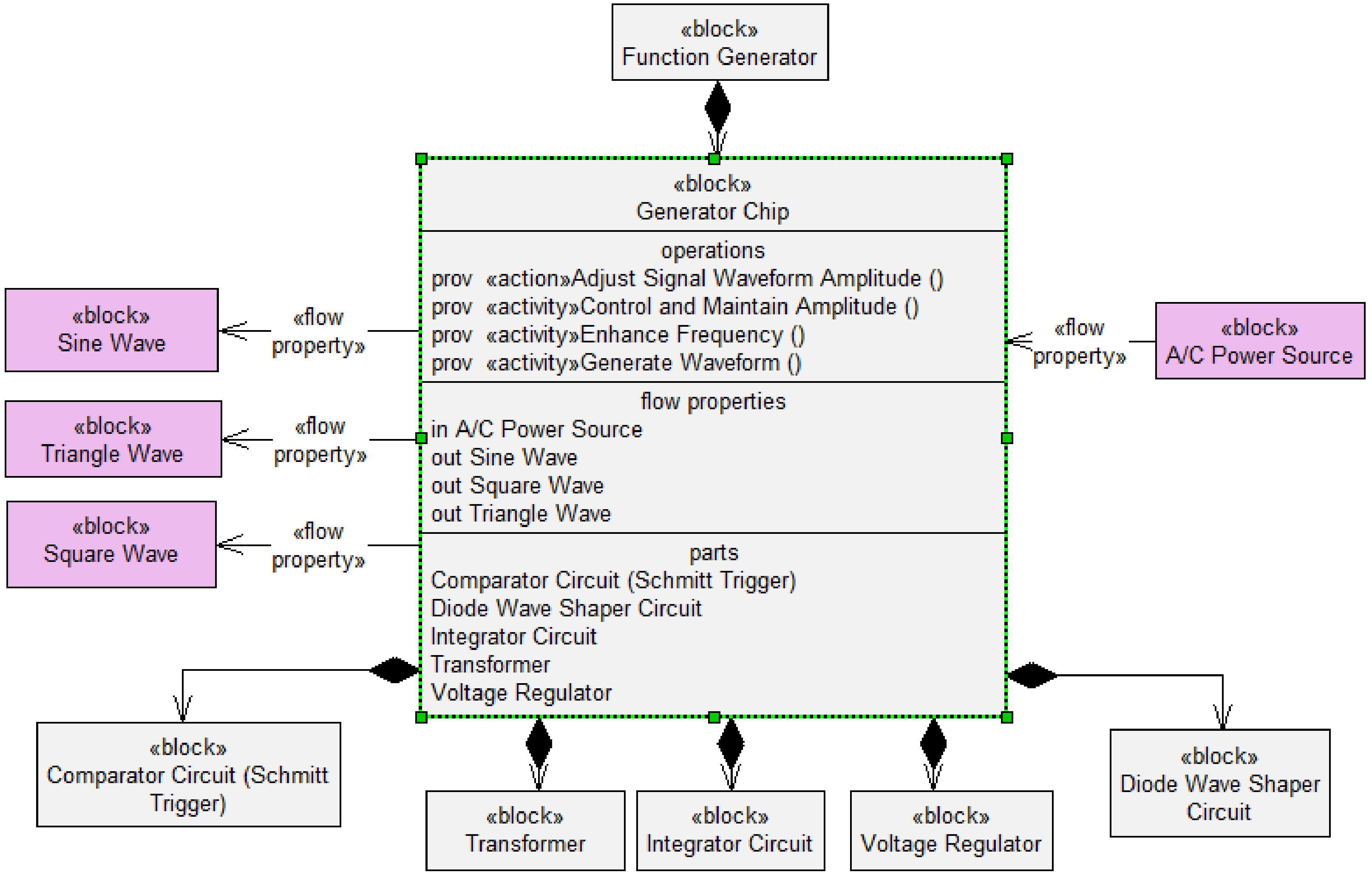


- Rectangle: A «block» element is a type of thing (system, subsystem, organization, report, etc)
- White Diamond: Reference / Shared Blocks on other end of path
- Black Diamond: Composed of Part Blocks on other end of path
- Hollow Triangle: specialized configuration/variant on other end of path
- Open Arrow head annotated with 'flow property': Specifies a flow element

3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

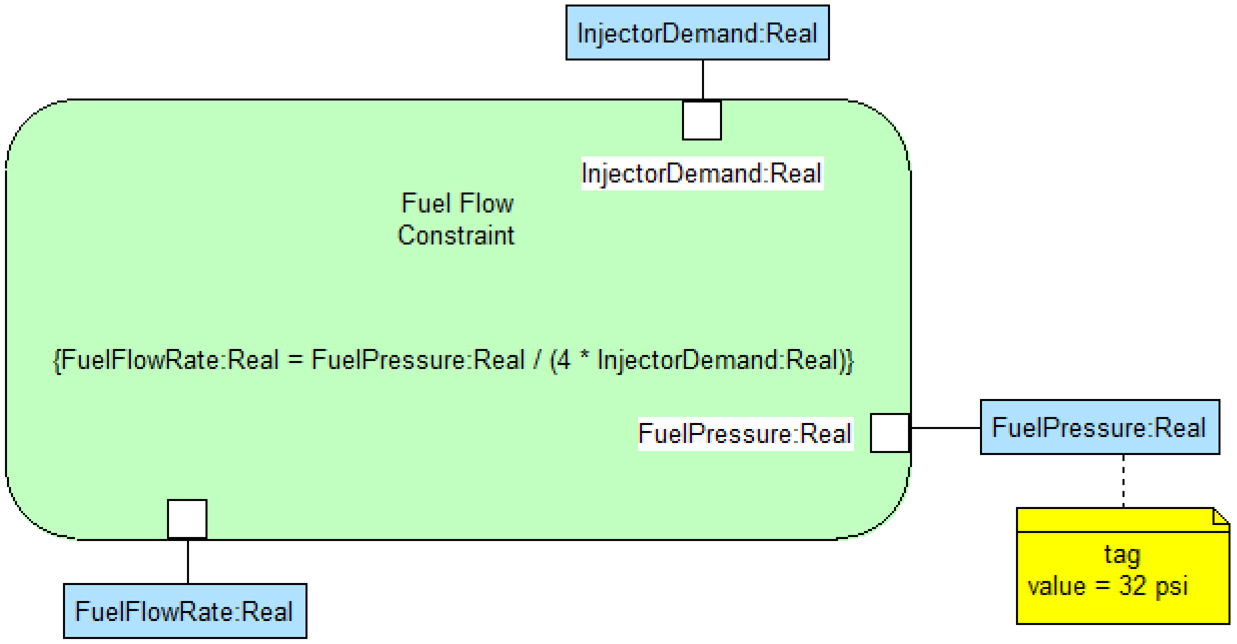
SysML Block Element Compartments


«block» Function Generator
«block» Generator Chip
operations
prov «action»Adjust Signal Waveform Amplitude ()
prov «activity»Control and Maintain Amplitude ()
prov «activity»Enhance Frequency ()
prov «activity»Generate Waveform ()
flow properties
in A/C Power Source
out Sine Wave
out Square Wave
out Triangle Wave
parts
Comparator Circuit (Schmitt Trigger)
Diode Wave Shaper Circuit
Integrator Circuit
Transformer
Voltage Regulator
«block» Sine Wave
«flow property»
«block» Triangle Wave
«flow property»
«block» Square Wave
«flow property»
«flow property»
«block» A/C Power Source
«block» Comparator Circuit (Schmitt Trigger)
«block» Transformer
«block» Integrator Circuit
«block» Voltage Regulator
«block» Diode Wave Shaper Circuit

Parametric Diagram (par) Used to Define Constraints

Parametric Diagram (par) – A Parametric Diagram show performance constraint equations, defined in constraint blocks, with a graphical mapping of each constraint equation parameter to a specific value property.

Parametric models thus focus on identifying the properties of the system that are critical to satisfying the identified performance requirements. An example is shown in the following Parametric diagram.

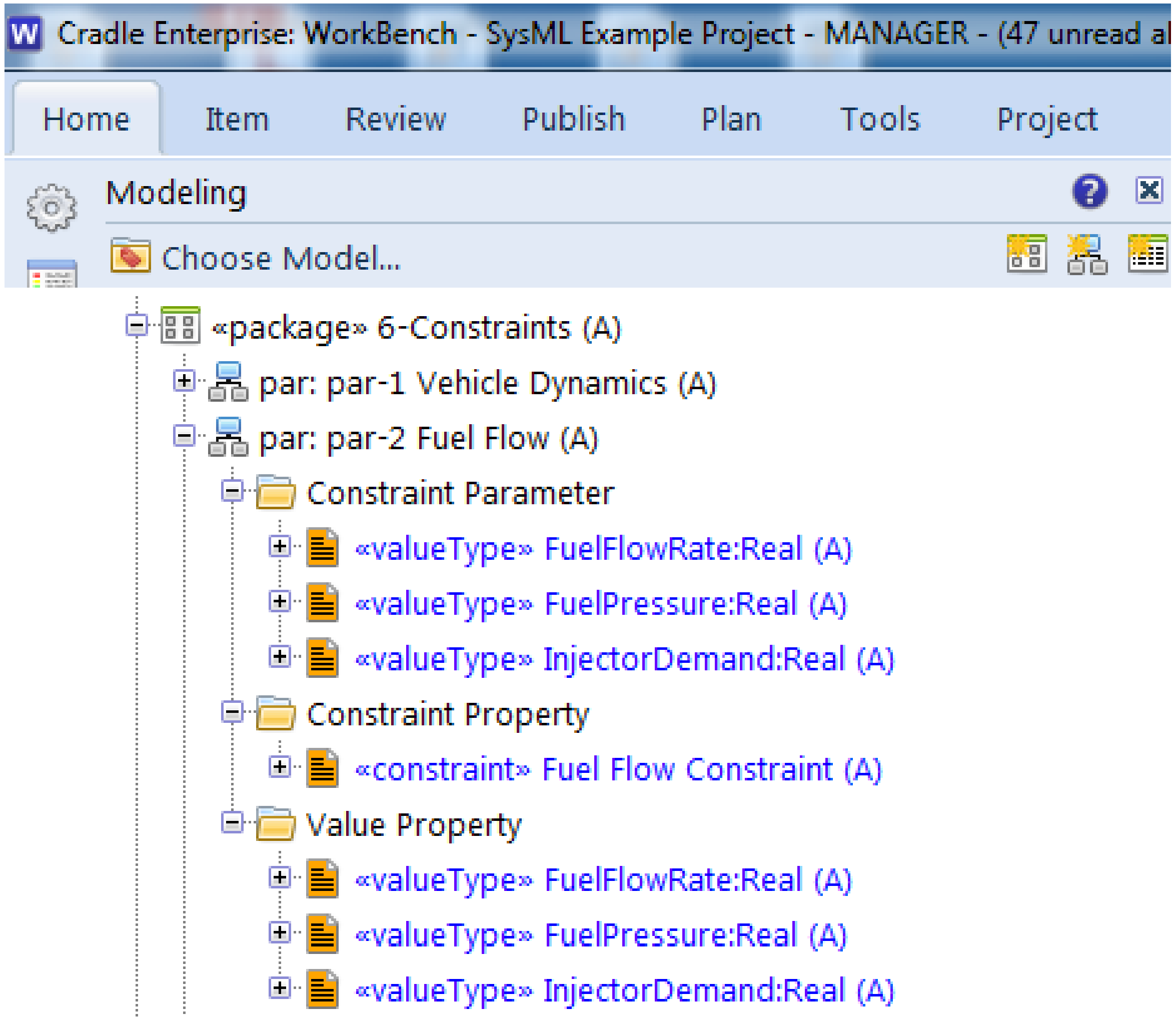


SysML Stereotypes Used on Sample par Diagram

«valueType»

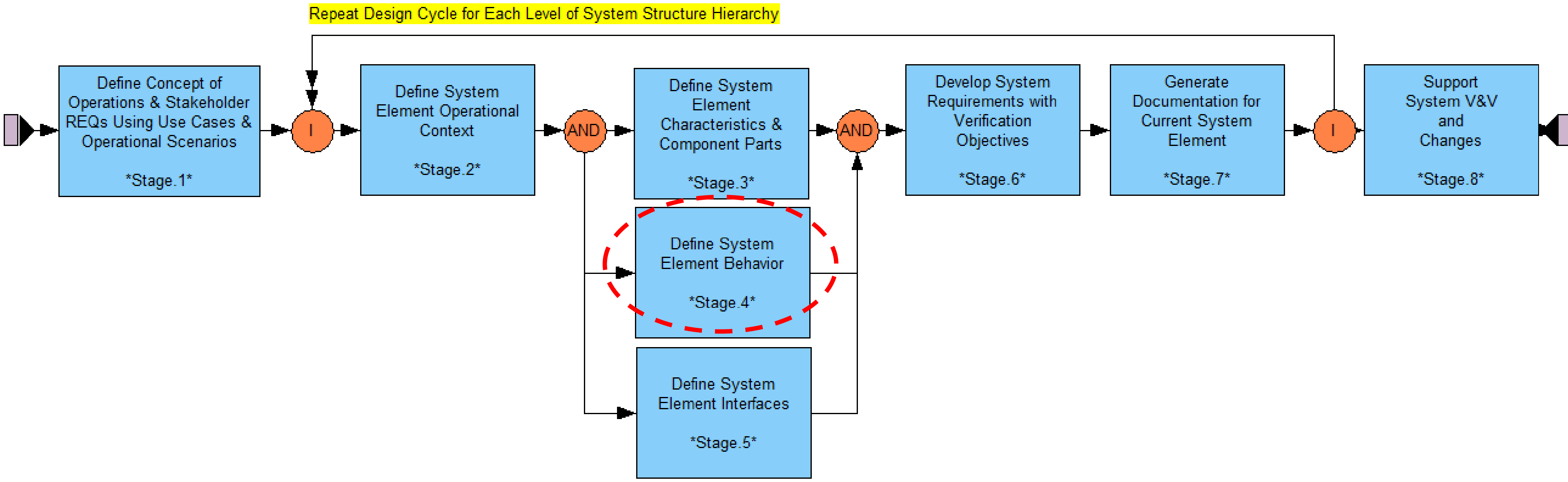
«constraint»

«valueType»


Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al
Home
Item
Review
Publish
Plan
Tools
Project
Modeling
Choose Model...

«package» 6-Constraints (A)
- par: par-1 Vehicle Dynamics (A)
- par: par-2 Fuel Flow (A)
  - Constraint Parameter
    - «valueType» FuelFlowRate:Real (A)
    - «valueType» FuelPressure:Real (A)
    - «valueType» InjectorDemand:Real (A)
  - Constraint Property
    - «constraint» Fuel Flow Constraint (A)
  - Value Property
    - «valueType» FuelFlowRate:Real (A)
    - «valueType» FuelPressure:Real (A)
    - «valueType» InjectorDemand:Real (A)

Stage 4 - Define System Element Behavior

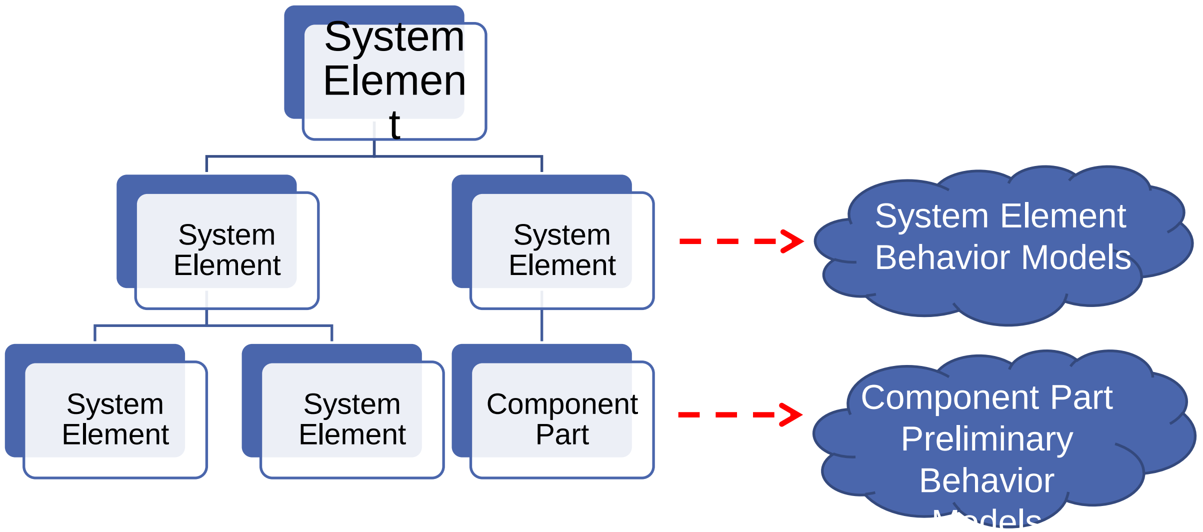

Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

Stage 4 - Define System Element Behavior

Define the behavior models that will ‘satisfy’ the behavioral requirements allocated to the System Element.

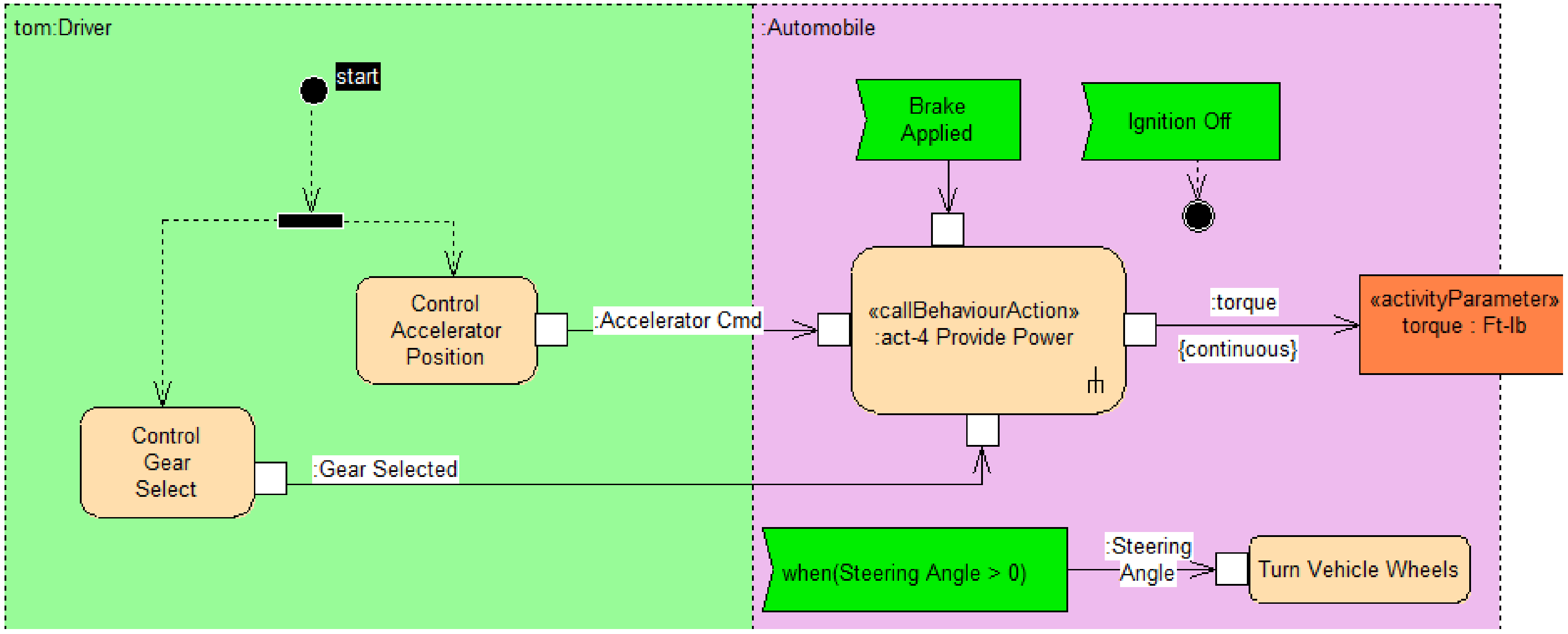
There are three kinds of SysML behavior diagrams: Activity, Sequence-Interaction, and State Machine diagrams. Which diagram-type to be used is based on the Systems Engineering Process

- Activity Diagrams (act) model functional flow-based behavior for a System Element
- Sequence-Interaction Diagrams (sd) illustrate the key events that occur between different System Elements
- State Machine diagrams (stm) specify the operating states for a System Element and the conditional transitions between States, and defines what behavior is performed when entering, dwelling in, and exiting a State.



Function Flow Behavior using SysML act

The Activity Diagram (act) is used to model the behavior of a System Element to show that the allocated behavioral requirements are satisfied. The activity diagram describes the transformation of inputs to outputs through a controlled time ordered sequence of actions (i.e., functions) to be performed by the System Elements.



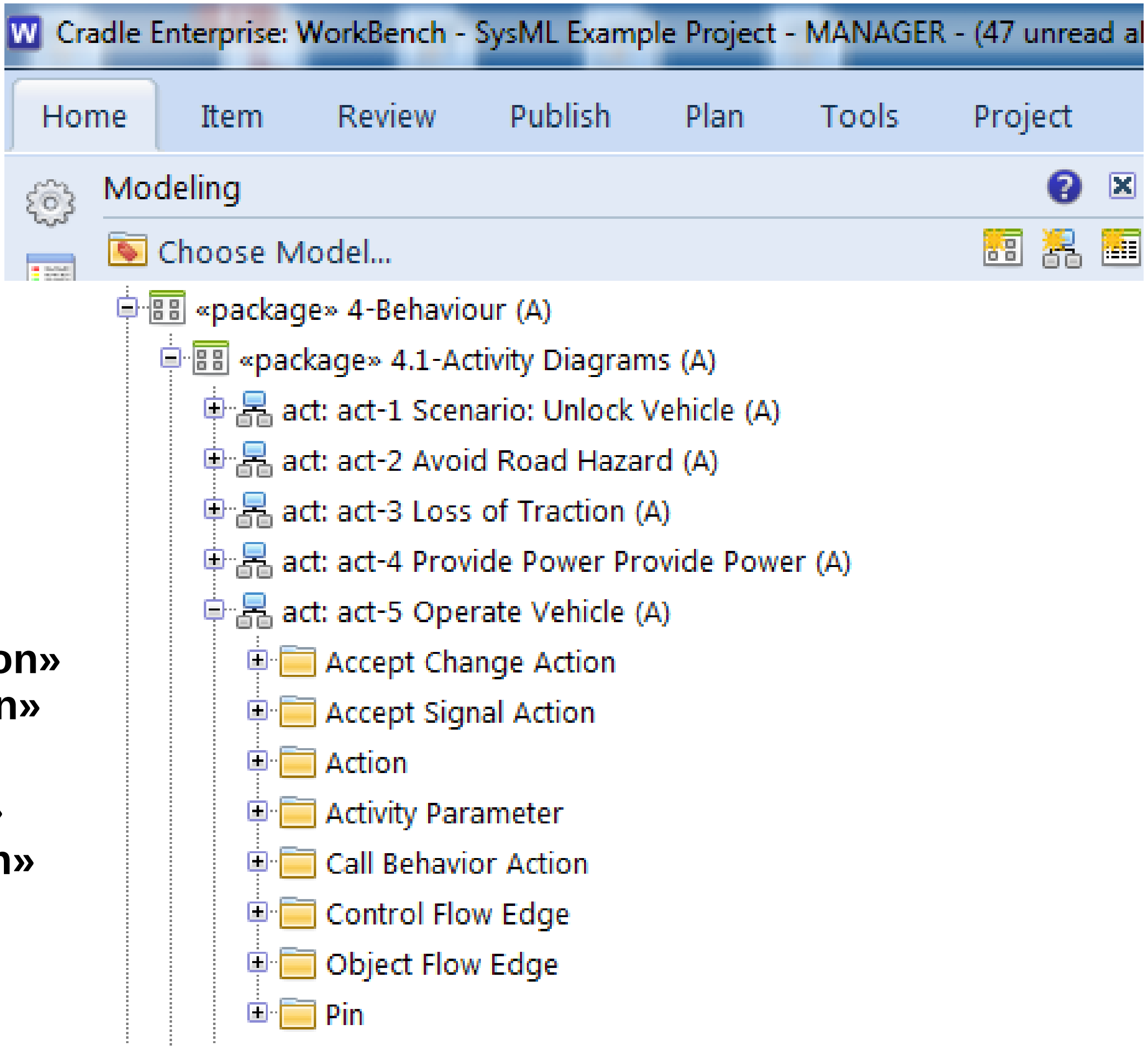
- Action: rounded corner rectangle
- Activity Parameter: rectangle
- Pin: small square on outside edge of Action
- Initial Node: solid black circle

- Fork: solid black bar
- Control Flow : dashed arrow
- Object Flow: solid arrow
- Accept Signal/Change Action: rectangle with triangle on left side

- Active Final Node: bulls-eye circle
- Performer (System Element): swim lanes

SysML Stereotypes Used on Sample act Diagram

«acceptChangeAction»
«acceptSignalAction»
«action»
«activityParameter»
«callBehaviorAction»
«controlFlowEdge»
«objectFlowEdge»
«pin»


Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

Home | Item | Review | Publish | Plan | Tools | Project

Modeling

Choose Model...

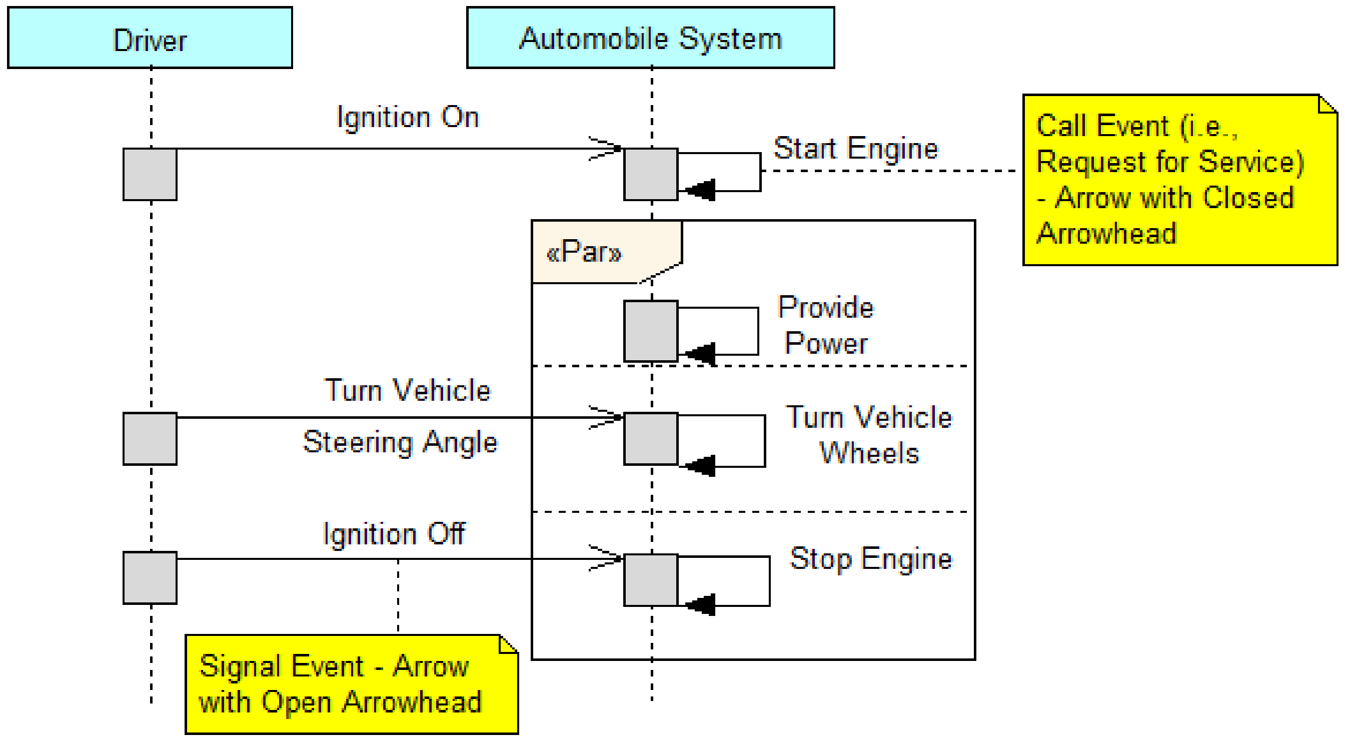
- «package» 4-Behaviour (A)
  - «package» 4.1-Activity Diagrams (A)
    - act: act-1 Scenario: Unlock Vehicle (A)
    - act: act-2 Avoid Road Hazard (A)
    - act: act-3 Loss of Traction (A)
    - act: act-4 Provide Power Provide Power (A)
    - act: act-5 Operate Vehicle (A)
      - Accept Change Action
      - Accept Signal Action
      - Action
      - Activity Parameter
      - Call Behavior Action
      - Control Flow Edge
      - Object Flow Edge
      - Pin

3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

Sequence-Interaction Behavior using SysML sd

A SysML Sequence-Interaction Diagram (sd) illustrates Message Events that get exchanged between System Elements and/or External Systems. The sequence diagram is particularly useful in modeling *service-oriented* operation request events and signal events.

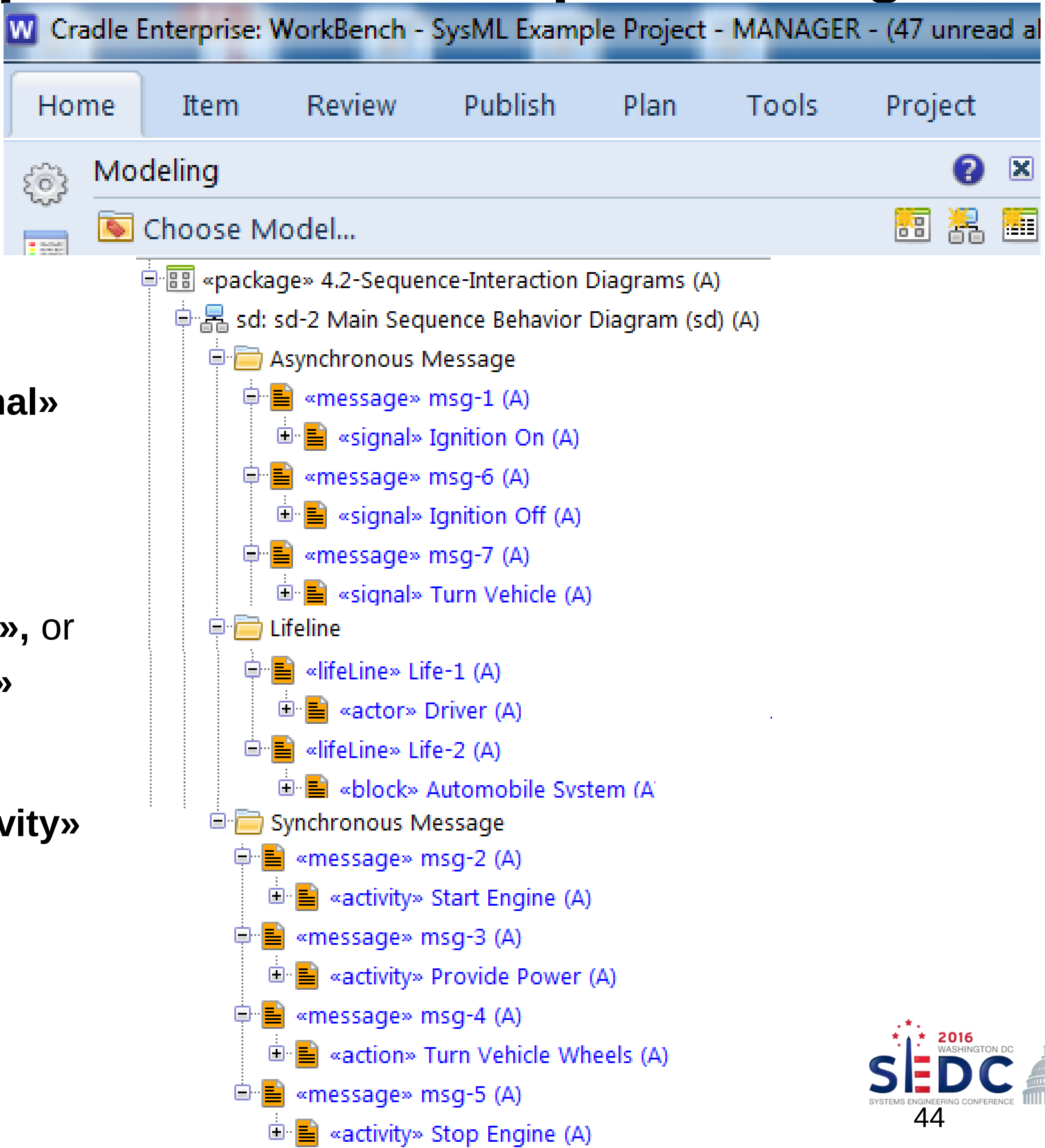


- Performer Lifeline: rectangle with dashed lifeline attached to bottom
- Execution Specification: rectangle overlaid on lifeline

- Signal Event Message: arrow with open arrowhead
- Call Event message: arrow with closed arrowhead

- Fragment: see through rectangle frame with title area in upper left corner

SysML Stereotypes Used on Sample sd Diagram



Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

- «package» 4.2-Sequence-Interaction Diagrams (A)
  - sd: sd-2 Main Sequence Behavior Diagram (sd) (A)
    - Asynchronous Message
      - «message» msg-1 (A)
        - «signal» Ignition On (A)
      - «message» msg-6 (A)
        - «signal» Ignition Off (A)
      - «message» msg-7 (A)
        - «signal» Turn Vehicle (A)
    - Lifeline
      - «lifeLine» Life-1 (A)
        - «actor» Driver (A)
      - «lifeLine» Life-2 (A)
        - «block» Automobile System (A
    - Synchronous Message
      - «message» msg-2 (A)
        - «activity» Start Engine (A)
      - «message» msg-3 (A)
        - «activity» Provide Power (A)
      - «message» msg-4 (A)
        - «action» Turn Vehicle Wheels (A)
      - «message» msg-5 (A)
        - «activity» Stop Engine (A)

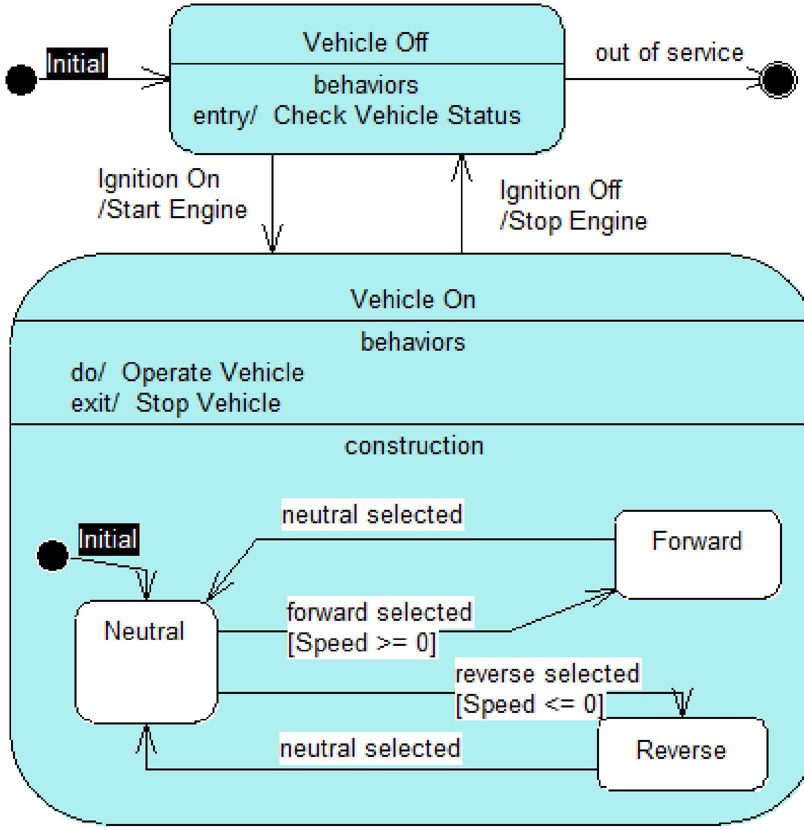
«message» references **«signal»**

«lifeLine» references **«block»,** or
«lifeLine» references **«actor»**

«message» references **«activity»**

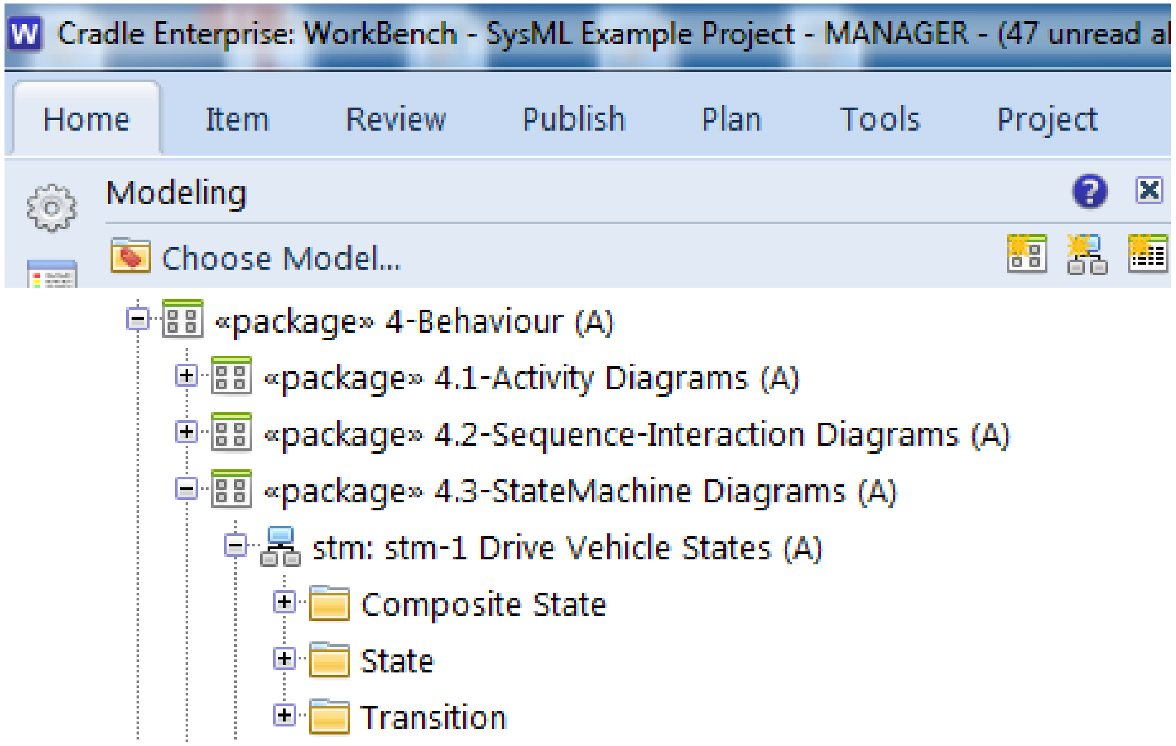
State Machine Behavior using SysML stm

The State Machine diagram (stm) identifies the operating states for a System Element and the conditional transitions between States, and what behavior is to be performed when entering, dwelling in, and exiting a State. There are four kinds of events that can cause a System Element to transition to another state: Signal Event, Call Event, Change Event, Time Event. The activity behaviors that can be performed by the State Machine is indicated by the "/" symbol in front of the Activity name.



- State: rounded corner rectangle
- Composite State: rounded corner rectangle with 'construction' compartment
- Initial State: solid black circle
- Final State: bulls-eye circle
- Transition: arrow with optional annotations
- Behavior to be performed: Forward slash followed by behavior name

SysML Stereotypes Used on Sample stm Diagram


Cradle Enterprise: WorkBench - SysML Example Project - MANAGER - (47 unread al

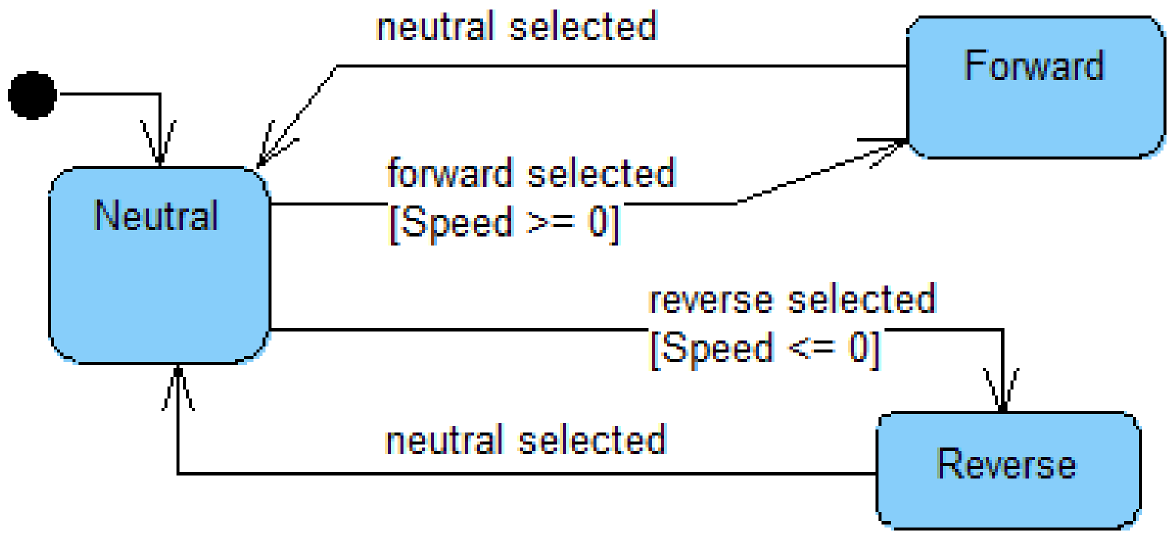
«package» 4-Behaviour (A)
«package» 4.1-Activity Diagrams (A)
«package» 4.2-Sequence-Interaction Diagrams (A)
«package» 4.3-StateMachine Diagrams (A)
stm: stm-1 Drive Vehicle States (A)
Composite State
State
Transition

«state»
«state»
«transition»

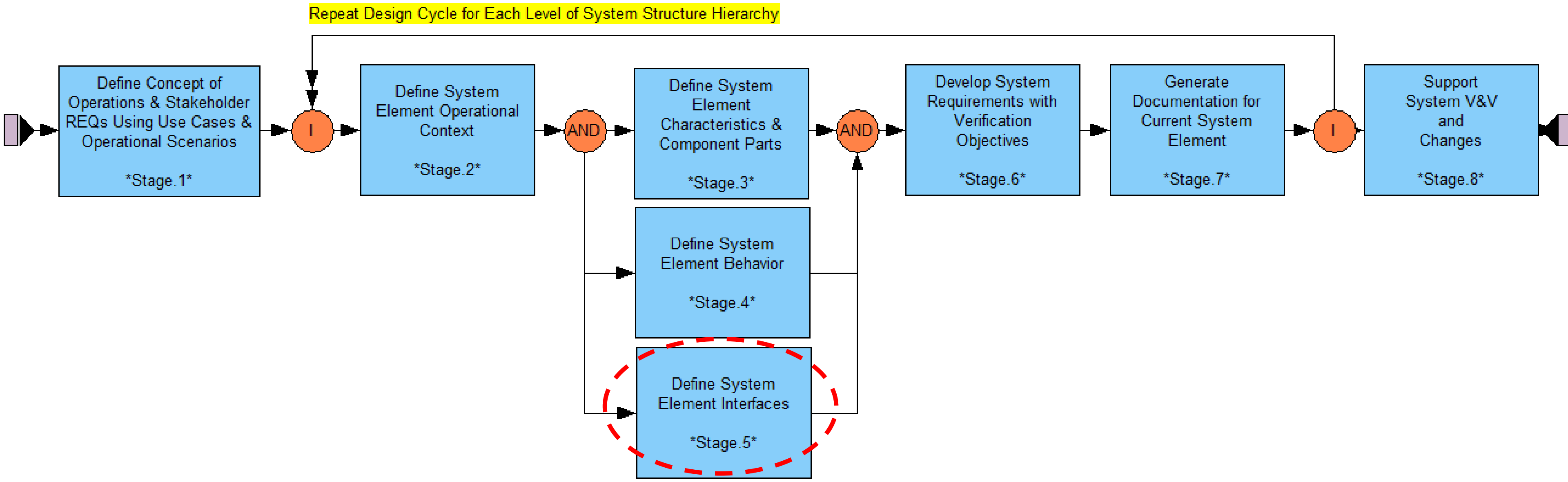
State Machine Behavior using SysML stm :2

A transition from one state to another is represented by an arrow symbol. The transition can be controlled by a ‘trigger event’ and/or a ‘guard expression’. The ‘trigger event’ can be a ‘signal event’, a ‘call event’, a ‘change event’, or a ‘time event’.

Signal Events (‘neutral selected’, ‘forward selected’, reverse selected’) that trigger transitions are illustrated in the following example.



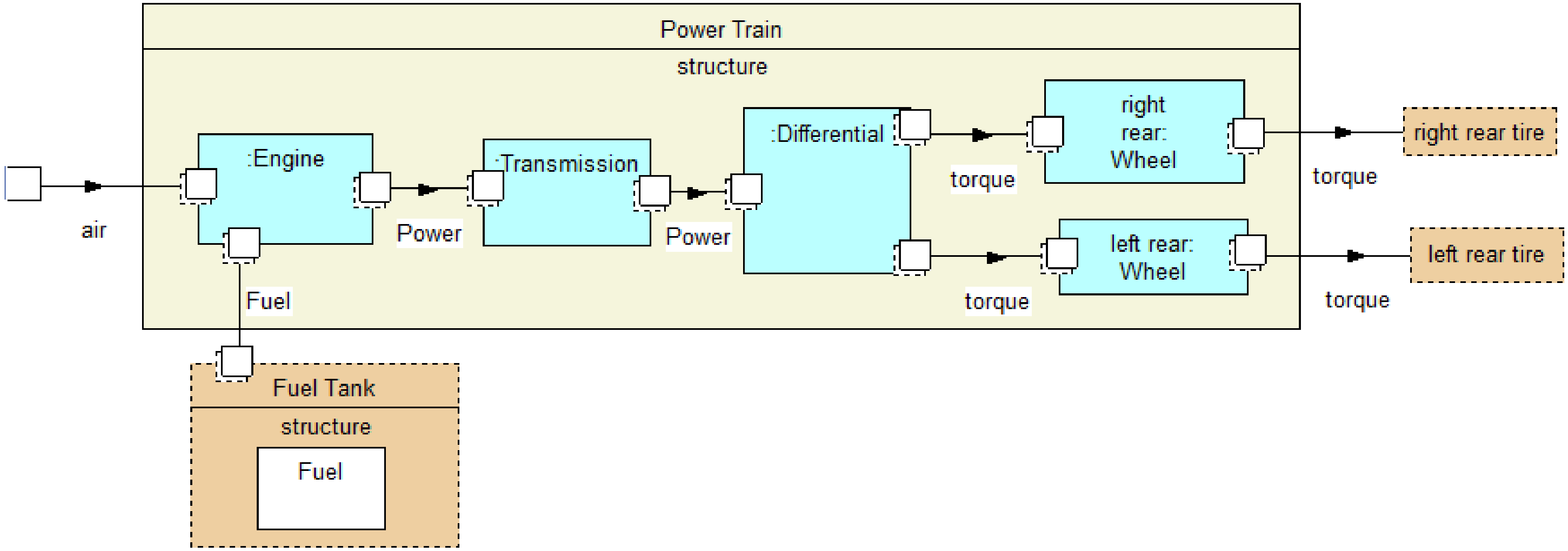
Stage 5 - Define System Element Interfaces


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

System Element Interfaces Using SysML bdd / ibd

The SysML Internal block diagram (ibd) is used to model the internal interconnections of the System Element's Component Parts and the interactions (i.e., interfacing information) between the Component Parts and the outside world. The interfacing information should be documented in the System Element's ICD.

The dashed rectangles represent external systems and the Item Flows (solid arrowhead

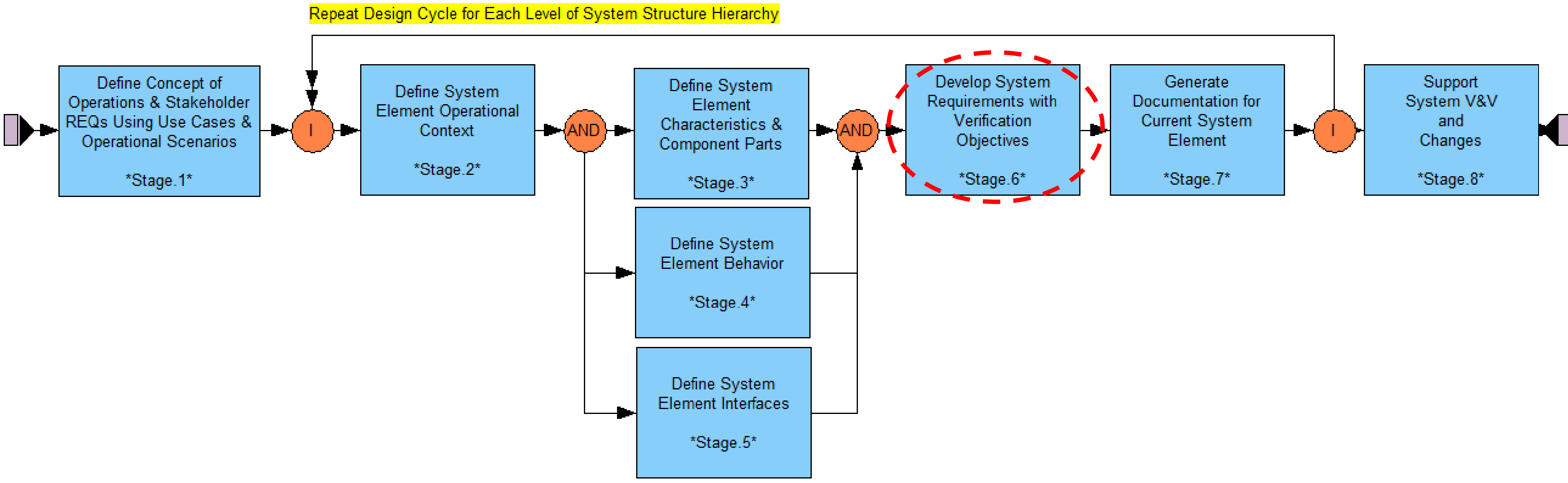


- Component Part: rectangle with solid boundary
- Referenced/Shared Element: rectangle with dashed boundary

- Proxy Port: small overlaid squares on the boundary of rectangle or diagram
- Full Port: small square on the boundary of rectangle or diagram

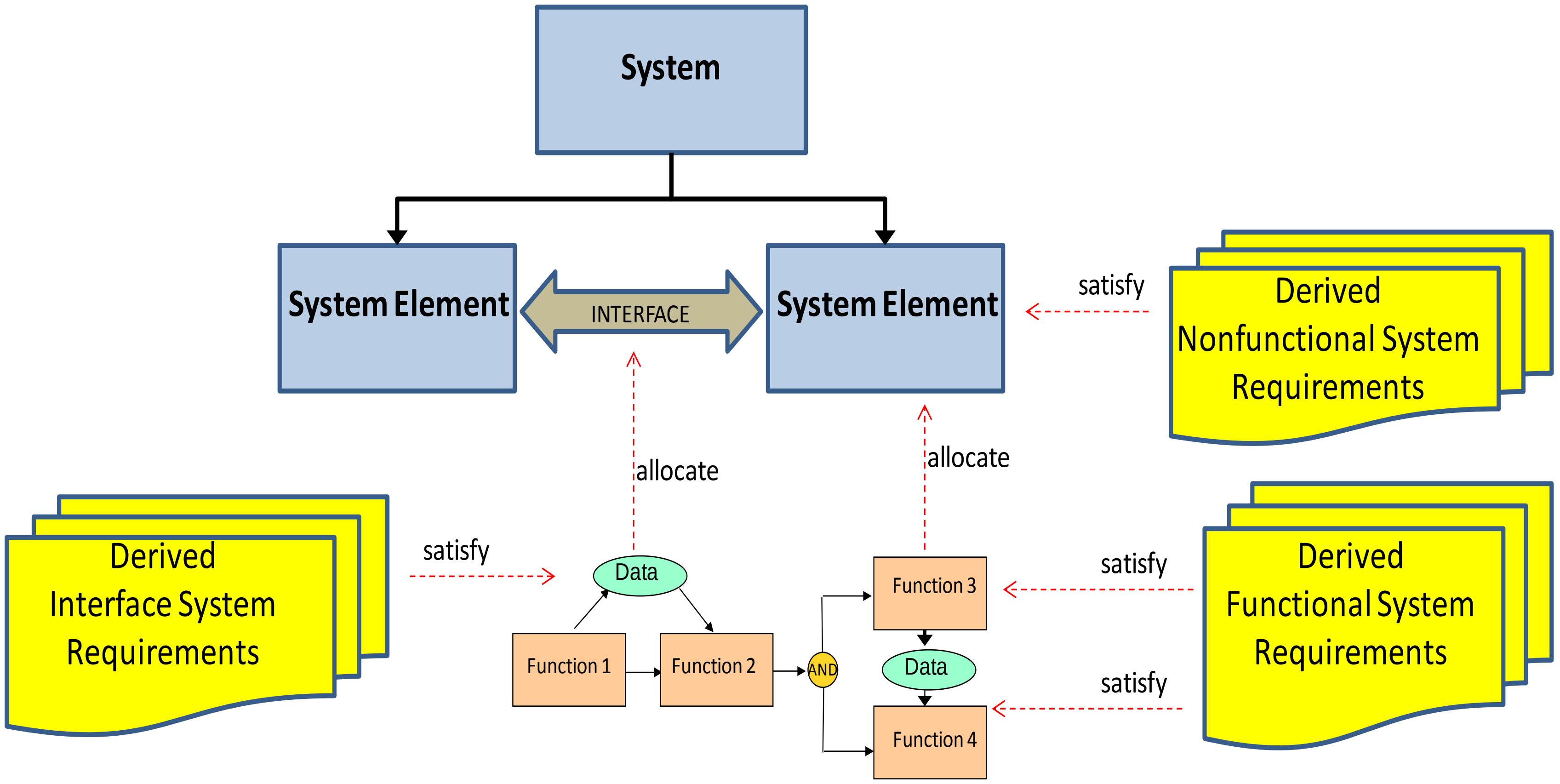
- Item Flow: solid arrowhead straddling connector line

Stage 6 - Develop System Requirements with Verification Objectives


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

Requirements & Traceability

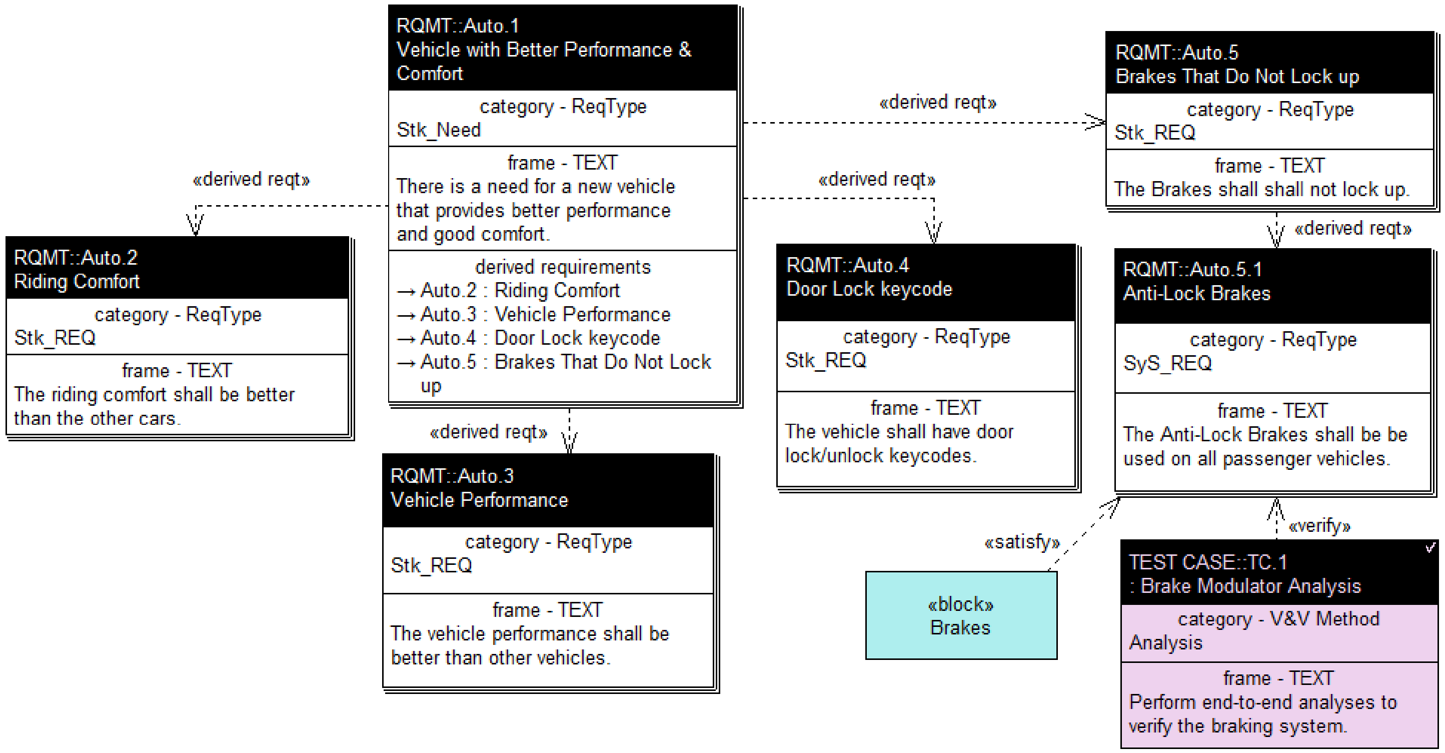
The system requirements derived in stages 3-5 are analyzed for clarity, completeness, consistency, traceability to the stakeholder or system requirements baselined at the end of the previous design cycle.



SysML Requirements Diagram (req)

Requirements Diagram (req) - Requirements Diagrams are used to graphically depict hierarchies of Stakeholder Needs, Requirements and Test Cases and show traceability to other model elements.

Requirements are usually displayed in table format rather than the Requirements Diagram because of the large number of requirements. In the following example, traceability from Stakeholder Needs to Stakeholder REQs to System REQs is shown. Traceability between the System Requirements and Verification / Test Cases is also shown.


RQMT::Auto.1
Vehicle with Better Performance & Comfort
category - ReqType
Stk_Need
frame - TEXT
There is a need for a new vehicle that provides better performance and good comfort.
derived requirements
→ Auto.2 : Riding Comfort
→ Auto.3 : Vehicle Performance
→ Auto.4 : Door Lock keycode
→ Auto.5 : Brakes That Do Not Lock up
«derived reqt»
RQMT::Auto.2
Riding Comfort
category - ReqType
Stk_REQ
frame - TEXT
The riding comfort shall be better than the other cars.
«derived reqt»
RQMT::Auto.3
Vehicle Performance
category - ReqType
Stk_REQ
frame - TEXT
The vehicle performance shall be better than other vehicles.
«derived reqt»
RQMT::Auto.4
Door Lock keycode
category - ReqType
Stk_REQ
frame - TEXT
The vehicle shall have door lock/unlock keycodes.
«derived reqt»
RQMT::Auto.5
Brakes That Do Not Lock up
category - ReqType
Stk_REQ
frame - TEXT
The Brakes shall shall not lock up.
«derived reqt»
RQMT::Auto.5.1
Anti-Lock Brakes
category - ReqType
SyS_REQ
frame - TEXT
The Anti-Lock Brakes shall be be used on all passenger vehicles.
«satisfy»
«block»
Brakes
«verify»
TEST CASE::TC.1
: Brake Modulator Analysis
category - V&V Method
Analysis
frame - TEXT
Perform end-to-end analyses to verify the braking system.

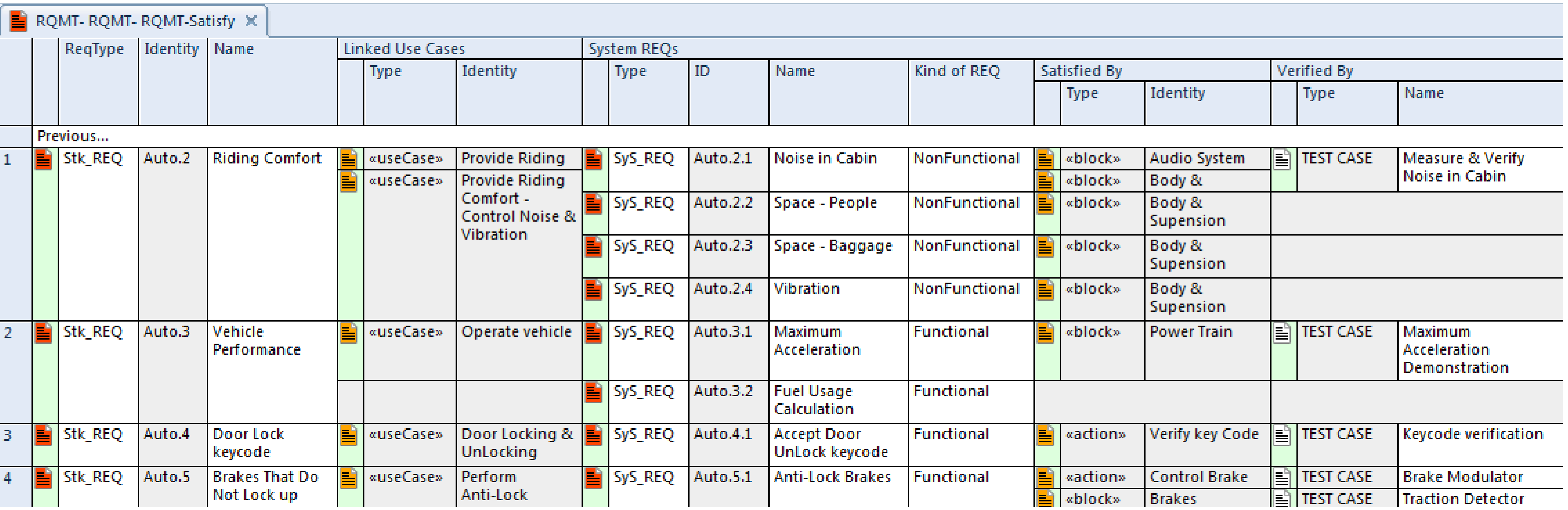
3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

Requirements to Modeling Elements Traceability

The ‘System Requirements Downward Traceability’ Query is run to show the results of cross reference linking the derived stakeholder requirements to the system requirements using the «derived reqt» relationship.

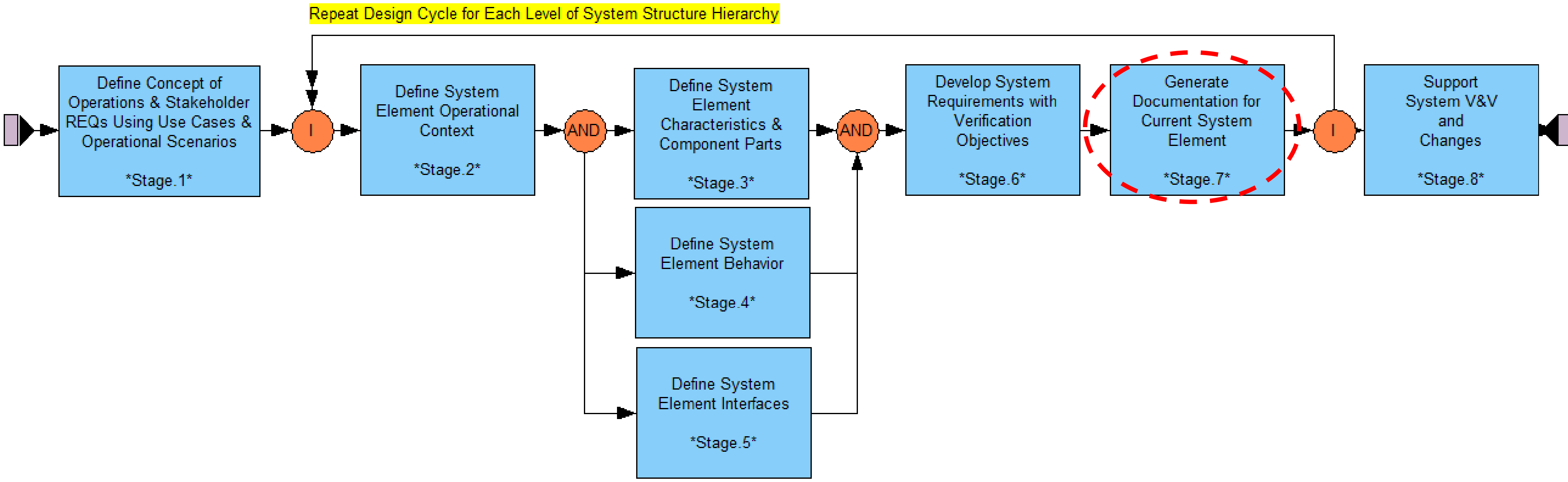
- The «satisfy» relationship is used to link the requirement items to the behavior elements («action» and «activity»)
- The «satisfy» relationship is used to link the requirement items to the structure elements («block»)
- The «verify» relationship is used to link the requirements to the verification / test cases elements



RQMT- RQMT- RQMT-Satisfy

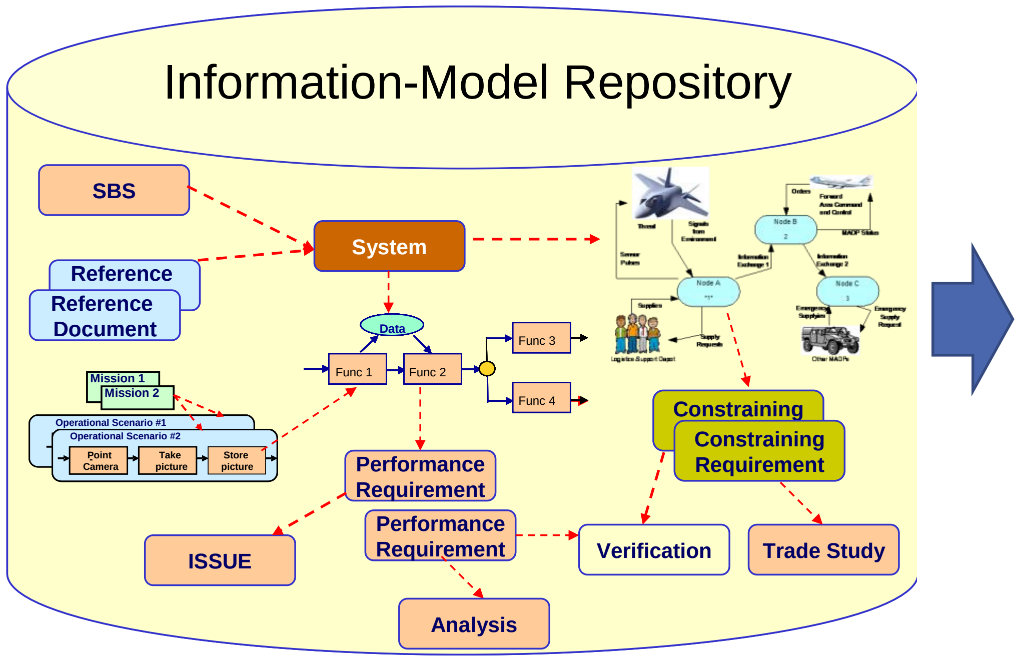
	ReqType	Identity	Name	Linked Use Cases		System REQs							
				Type	Identity	Type	ID	Name	Kind of REQ	Satisfied By		Verified By	
										Type	Identity	Type	Name
	Previous...												
1	Stk_REQ	Auto.2	Riding Comfort	«useCase»	Provide Riding	SyS_REQ	Auto.2.1	Noise in Cabin	NonFunctional	«block»	Audio System	TEST CASE	Measure & Verify Noise in Cabin
				«useCase»	Provide Riding Comfort - Control Noise & Vibration					«block»	Body &		
						SyS_REQ	Auto.2.2	Space - People	NonFunctional	«block»	Body & Supension		
						SyS_REQ	Auto.2.3	Space - Baggage	NonFunctional	«block»	Body & Supension		
						SyS_REQ	Auto.2.4	Vibration	NonFunctional	«block»	Body & Supension		
2	Stk_REQ	Auto.3	Vehicle Performance	«useCase»	Operate vehicle	SyS_REQ	Auto.3.1	Maximum Acceleration	Functional	«block»	Power Train	TEST CASE	Maximum Acceleration Demonstration
						SyS_REQ	Auto.3.2	Fuel Usage Calculation	Functional				
3	Stk_REQ	Auto.4	Door Lock keycode	«useCase»	Door Locking & UnLocking	SyS_REQ	Auto.4.1	Accept Door UnLock keycode	Functional	«action»	Verify key Code	TEST CASE	Keycode verification
4	Stk_REQ	Auto.5	Brakes That Do Not Lock up	«useCase»	Perform Anti-Lock	SyS_REQ	Auto.5.1	Anti-Lock Brakes	Functional	«action»	Control Brake	TEST CASE	Brake Modulator
										«block»	Brakes	TEST CASE	Traction Detector

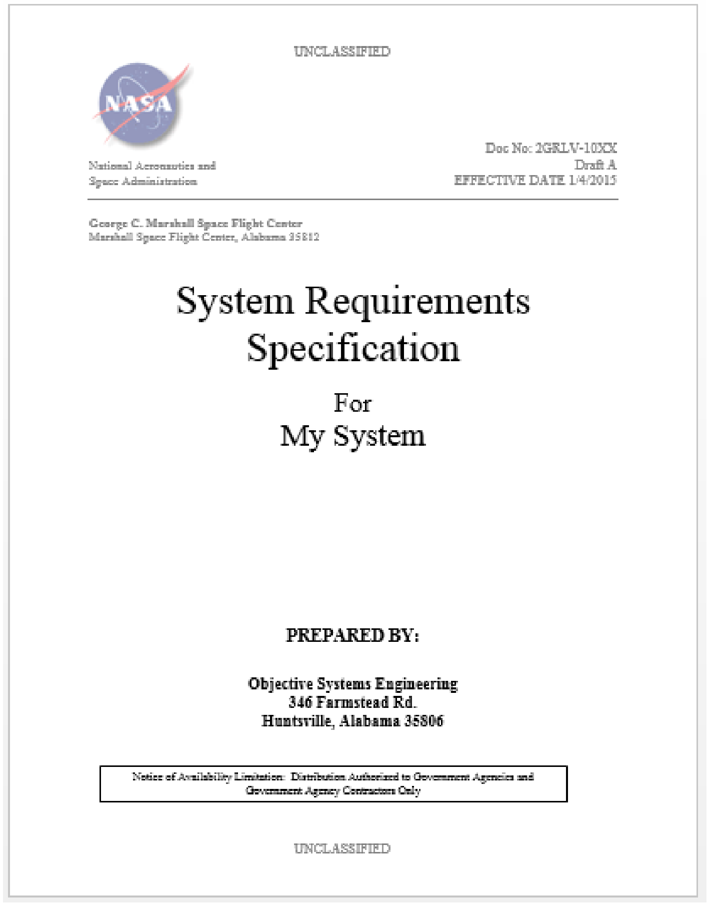
Stage 7 - Generate Documentation for Current System Element


Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

Automatic Document Generation

Automatic generation of documents from the Information-Model using pre-defined templates reduce the cost of creating the documents.




UNCLASSIFIED

National Aeronautics and
Space Administration

Doc No: 2GRLV-10XX
Draft A
EFFECTIVE DATE 1/4/2015

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

System Requirements
Specification

For
My System

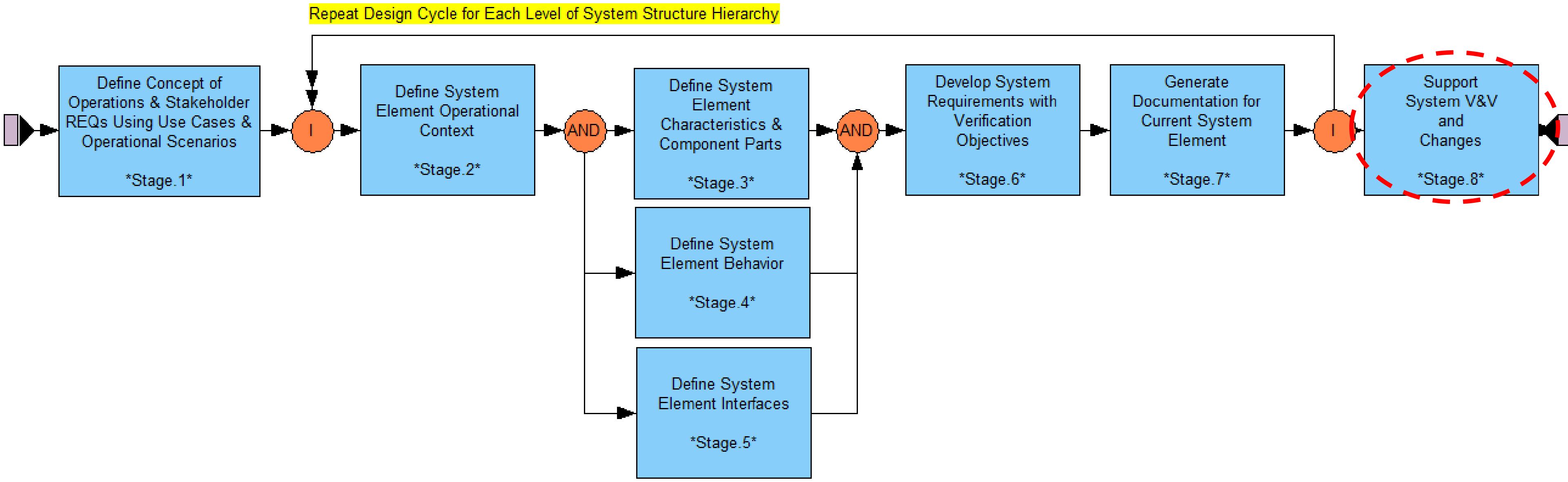
PREPARED BY:

Objective Systems Engineering
346 Farmstead Rd.
Huntsville, Alabama 35806

Notice of Availability Limitation: Distribution Authorized to Government Agencies and Government Agency Contractors Only

UNCLASSIFIED

Stage 8 - Support System V&V and Changes

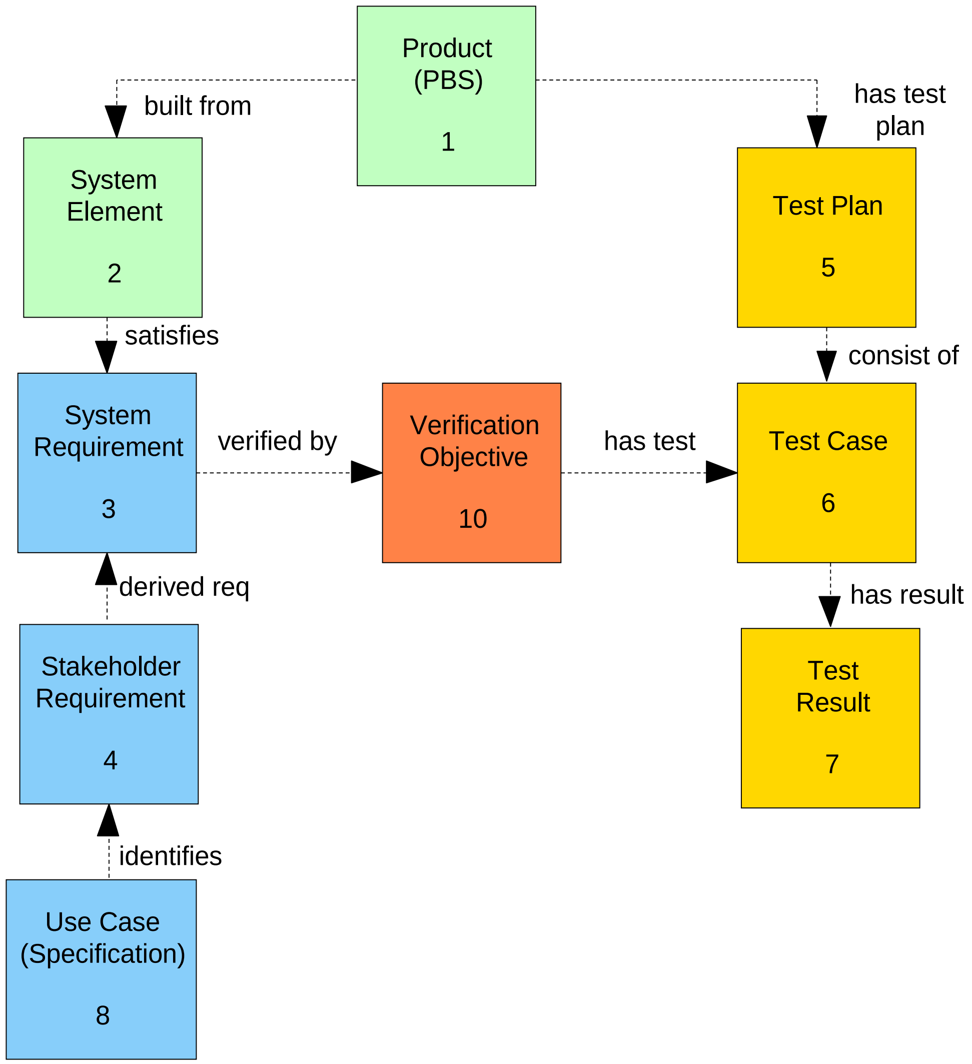

Repeat Design Cycle for Each Level of System Structure Hierarchy
Define Concept of Operations & Stakeholder REQs Using Use Cases & Operational Scenarios *Stage.1*
I
Define System Element Operational Context *Stage.2*
AND
Define System Element Characteristics & Component Parts *Stage.3*
Define System Element Behavior *Stage.4*
Define System Element Interfaces *Stage.5*
AND
Develop System Requirements with Verification Objectives *Stage.6*
Generate Documentation for Current System Element *Stage.7*
I
Support System V&V and Changes *Stage.8*

3SL

2016 WASHINGTON DC SEDC SYSTEMS ENGINEERING CONFERENCE

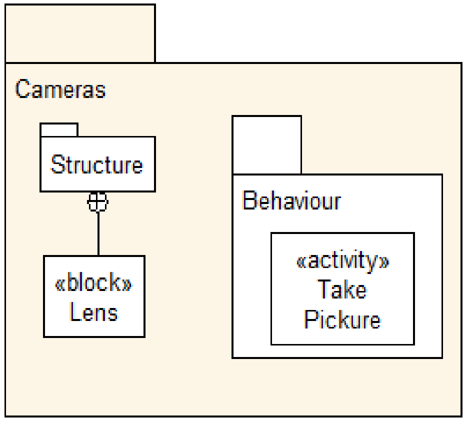
Requirements Verification Traceability

Verification and test planning activities should be performed for the newly created system requirements and appropriate traceability established.

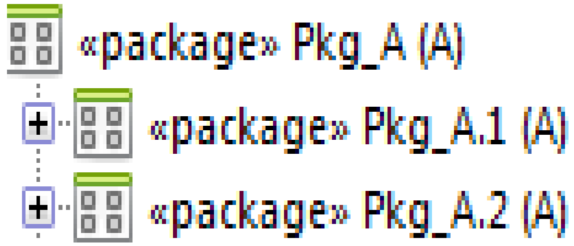


Package Diagrams (pkg)

Package Diagram (pkg) – A Package diagram is used to partition the modeling elements into logical groupings that describe a domain of interest. A hierarchy of packages can be defined in the Modeling Sidebar and in a Package Diagram. A «package» is like a folder. The following example shows elements contained in a package and nested packages.



The modeling browser is the primary place where nested packages are created and used. The following example shows nested packages.



Summary

Is Model Based Systems Engineering (MBSE) something new? No, MBSE methods and techniques have been individually practiced by many good engineers and analyst over the years.

The problem has been that project management has been obsessed with 'deliverable documents' rather than delivering an engineering Information-model that can be used to support analyses, end-to-end traceability, and automatically produce the deliverable documents from the Information-Model.

- The engineering data-model usually consists of entities, attributes, relationships, and diagrams that specify the system architecture.
- Remember a diagram (graphical model) is worth a thousand words. These diagrams aid in communicating ideas/concepts among project personnel and the customer.
- What is new is the SysML modeling language. It should help with communication issues by having a common way to describe system architectures and traceability.

Please contact me at loyd.baker@threesl.com to discuss MBSE processes and Learn SysML using Cradle or learn SysML independent of a tool.