COLLEGE of CHARLESTON

1770

Systems Engineering: Design and Development

ENGR 387

Agenda

- Systems Modeling Language (SysML)
- SysML Diagrams
  - Taxonomy
  - Overview
  - Format
  - Header Format
  - Examples
- Namespace
- Elements of Definition
- Elements of Usage
- Model Conventions and Standards
- Model Organization
- Summary
- References

COLLEGE of CHARLESTON

Systems Modeling Languate (SysML)

- The Object Management Group Systems Modeling Language (OMG SysML), simply stated as SysML, <u>is a general-purpose modeling language for systems engineering applications</u>

- SysML supports the specification, analysis, design, verification, and validation of a broad range of systems

- SysML consists of Abstract Syntax and Concrete Syntax:
    - <u>Abstract Syntax</u> is the set of <u>rules</u> that identify what you can and cannot do
    - <u>Concrete Syntax</u> is the set of <u>notations</u> that you are allowed to use on diagrams

Systems Modeling Languate (SysML)

- SysML Origin
  - The Unified Modeling Language (UML) was designed (1994-1995) as a general-purpose modeling language in the field of software engineering.
  - SysML is defined as an extension of a subset of the UML 2 standard.

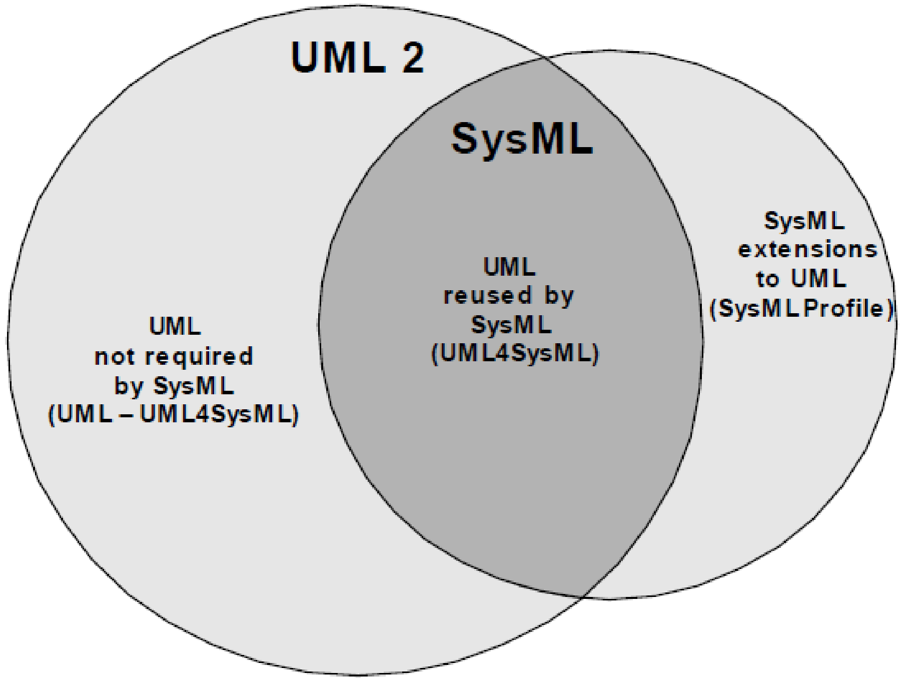
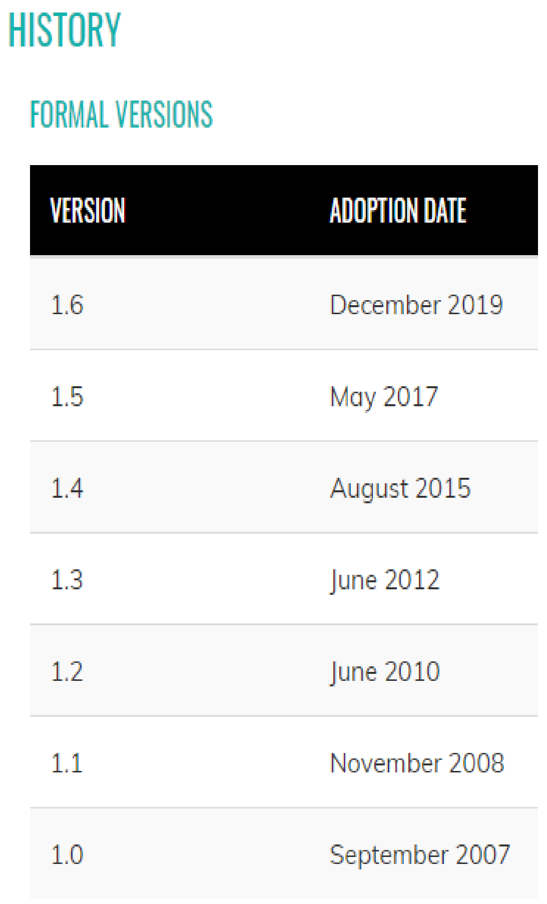

UML 2
SysML
UML not required by SysML (UML – UML4SysML)
UML reused by SysML (UML4SysML)
SysML extensions to UML (SysML Profile)

Figure 4.1 - Overview of SysML/UML Interrelationship

Systems Modeling Languate (SysML)

- SysML Versions:
  - Latest version is 1.6 (Dec 2019)
  - OCSMP exams are based on SysML v1.2
  - SysML v1.2 uses Standard and Flow Ports
  - SysML v1.3 introduced Full and Proxy Ports

- Notes:
  - The SysML specification contains advanced concepts not necessary for Level 1 or 2 exams!
  - Cameo Systems Modeler (CSM):
    - Compliant with SysML v1.3 or newer
    - Does not match the SysML spec exactly!!
    - May allow modelers to do things that are not allowed by SysML spec!
    - Service pack updates are NOT tied to SysML version updates!



HISTORY

FORMAL VERSIONS

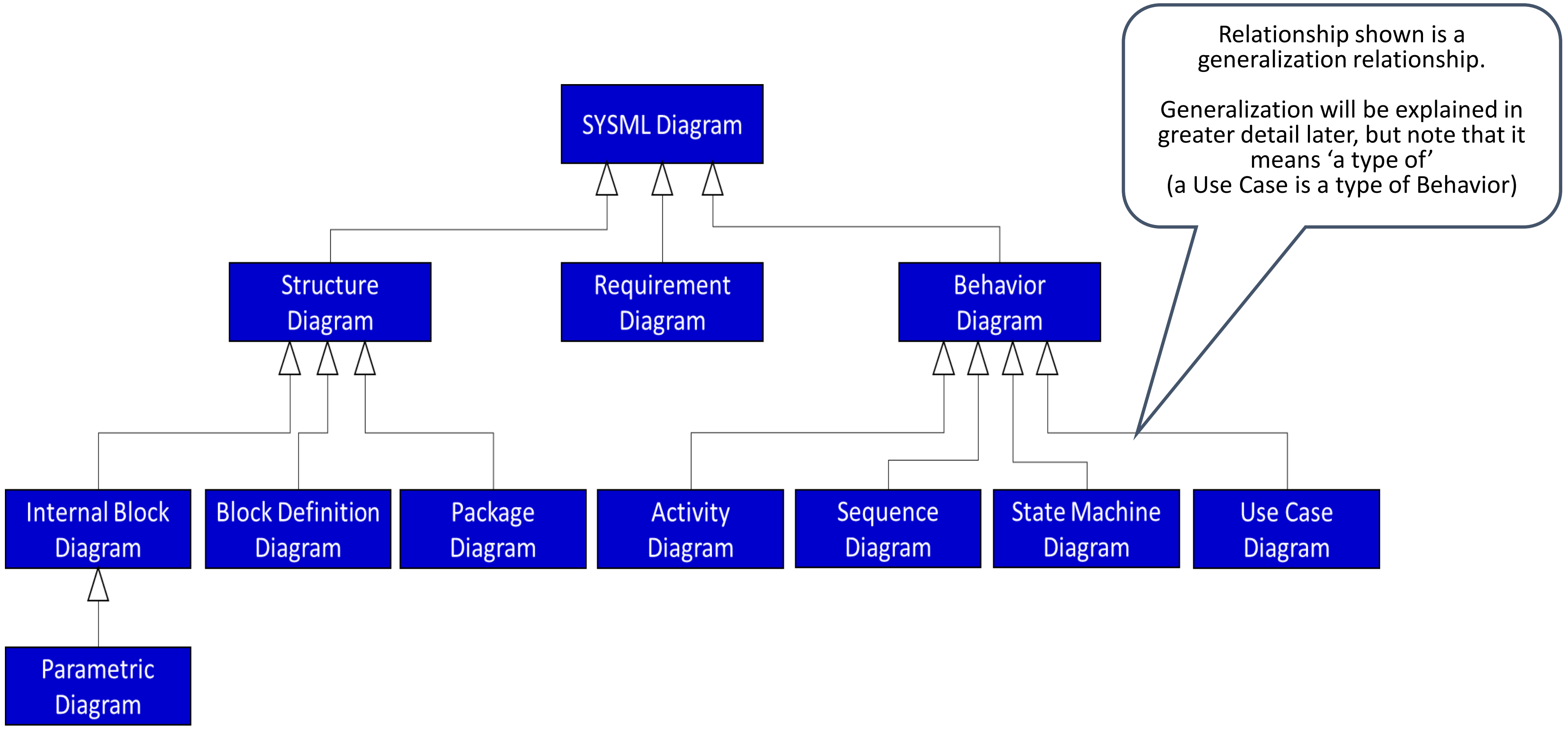
VERSION	ADOPTION DATE
1.6	December 2019
1.5	May 2017
1.4	August 2015
1.3	June 2012
1.2	June 2010
1.1	November 2008
1.0	September 2007

COLLEGE *of* CHARLESTON

SysML Diagrams

Taxonomy

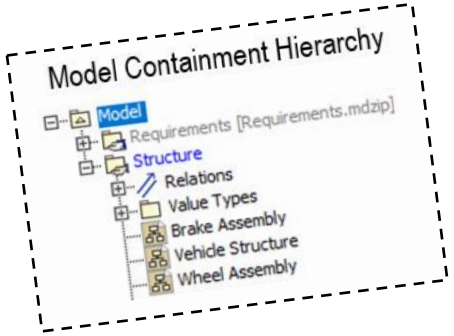
- The nine (9) SysML diagram kinds and their relationships to each other are shown below



COLLEGE of CHARLESTON

SysML Diagrams

Overview

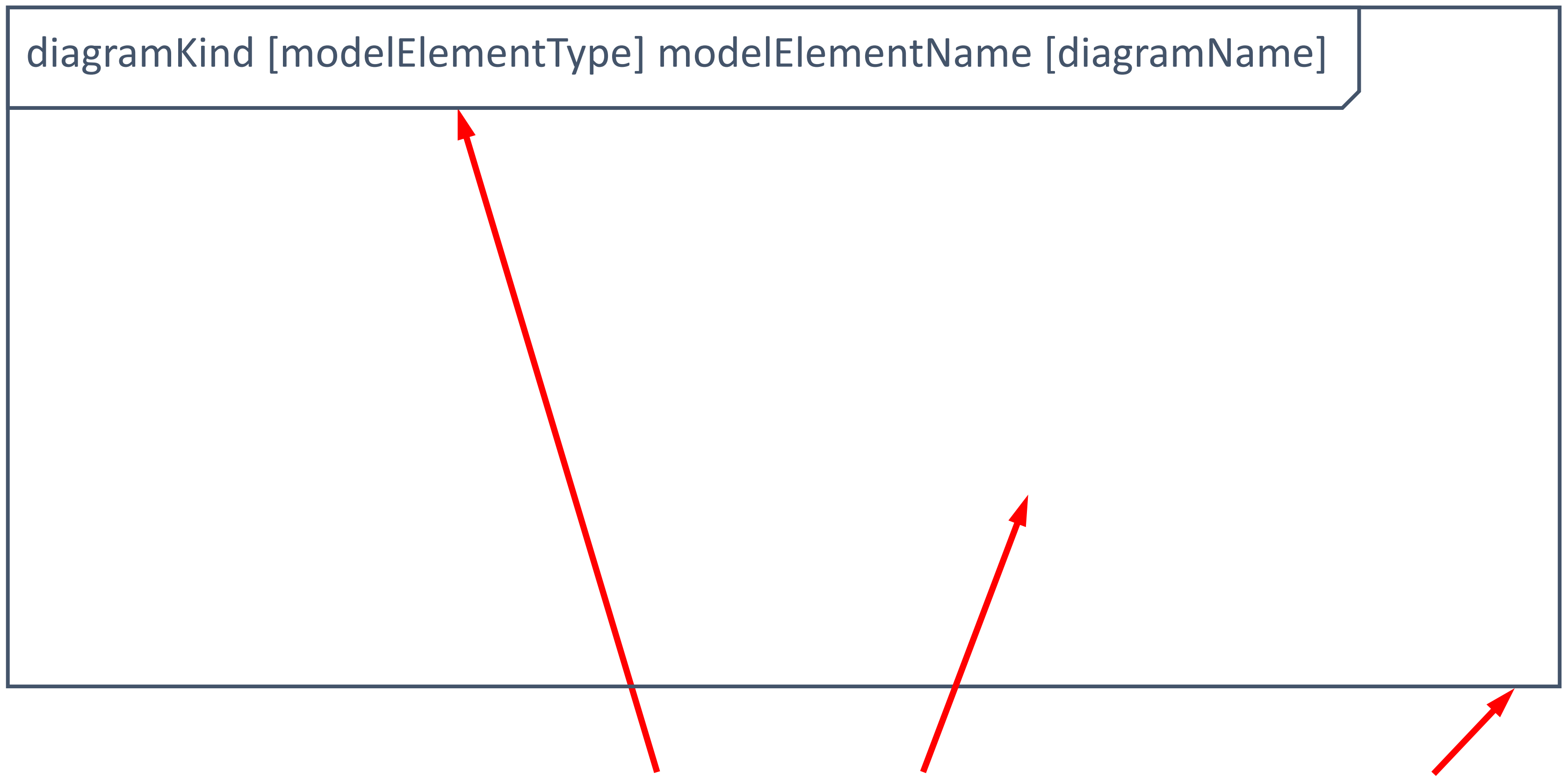


- What are SysML diagrams?
  - Diagrams are elements in your model
  - Diagrams are views of the model
  - The model itself is generally shown as elements within a hierarchical structure of nested packages (often referenced as the CSM term: Containment Tree)
  - A diagram is a <u>partial</u> graphical representation of the model that is generated to satisfy a stakeholder’s request
  - A diagram can also be a <u>partial</u> graphical representation of the model element that the diagram represents
  - An item can exist in your model without being shown on any diagrams
  - A set of diagrams does not need to completely cover the model
  - Deleting an item from a diagram does not remove it from the model
  - Deleting a diagram does not affect the elements, or the relationships between elements, that were displayed on that diagram

COLLEGE *of* CHARLESTON

SysML Diagrams

Format


diagramKind [modelElementType] modelElementName [diagramName]

- Each diagram has a header, a contents area, and a frame
  - Note that the header *commonly* includes all four pieces of information, but that book and exam diagrams may not necessarily include everything!

SysML Diagrams

Header Format

diagramKind [modelElementType] modelElementName [diagramName]

- diagramKind
  - SysML-defined acronym identifying the kind of diagram shown
    - activity diagram (act)
    - block definition diagram (bdd)
    - internal block diagram (ibd)
    - package diagram (pkg)
    - parametric diagram (par)
    - requirement diagram (req)
    - sequence diagram (sd)
    - state machine diagram (stm)
    - use case diagram (uc)

Example

bdd [Block] Vehicle [Powertrain Subsystem Components]

COLLEGE of CHARLESTON

SysML Diagrams

Header Format Cont.

diagramKind [modelElementType] modelElementName [diagramName]

- [modelElementType]
  - Remember that diagrams are model elements and they have a name and a type.
  - modelElementType is determined by the element that the diagram represents.
  - The most common element types per diagram kind are as follows:
    - act: activity
    - bdd: block, constraintBlock, package, model, modelLibrary, view
    - ibd: block
    - pkg: package, model, modelLibrary, view, profile
    - par: block, constraintBlock
    - req: package, model, modelLibrary, view, requirement
    - sd: interaction
    - stm: stateMachine
    - uc: package, model, modelLibrary, view

Example

bdd [Block] Vehicle [Powertrain Subsystem Components]

SysML Diagrams

Header Format Cont.

diagramKind [modelElementType] modelElementName [diagramName]

- [modelElementName]
  - Remember that diagrams are model elements and they have a name and a type.
  - modelElementName is the name of the model element that the diagram represents

Example

bdd [Block] Vehicle [Powertrain Subsystem Components]

COLLEGE of CHARLESTON
1770

SysML Diagrams

Header Format Cont.

diagramKind [modelElementType] modelElementName [diagramName]

- [diagramName]
  - This can be anything you would like, but should be a brief name to provide the reader some knowledge on what they should expect to see in that diagram.

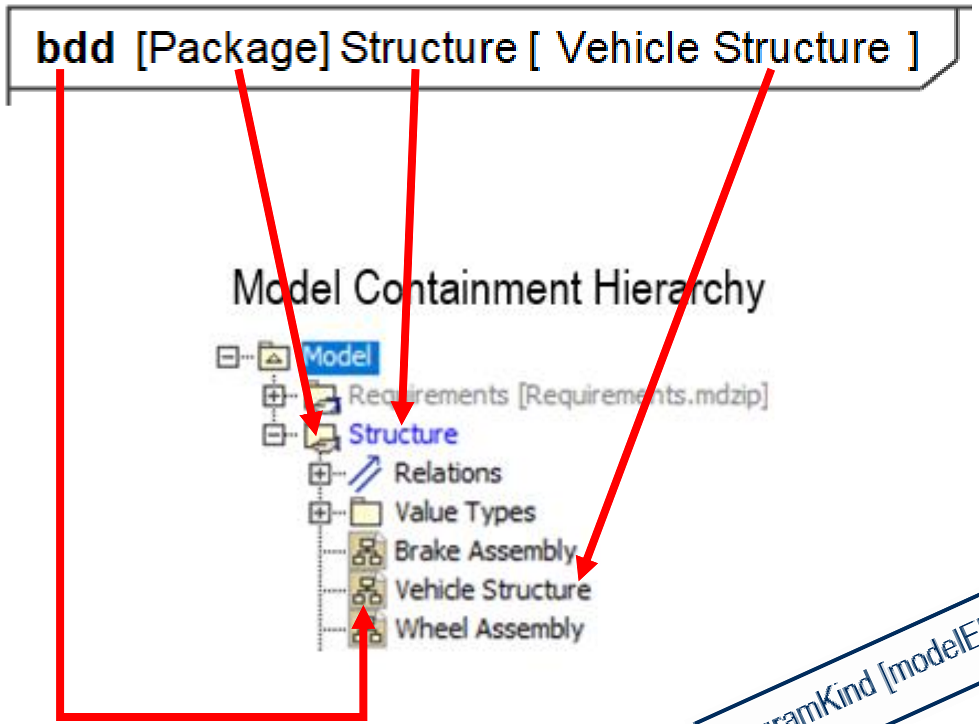
Example

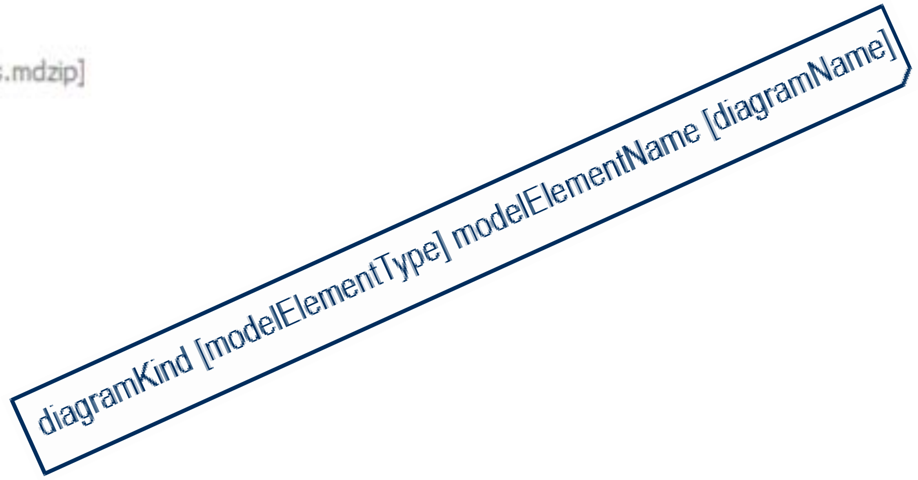
bdd [Block] Vehicle [Powertrain Subsystem Components]

SysML Diagrams

Example Header #1

- This diagram is a 'bdd' of the 'Structure' 'Package' and it's purpose is to illustrate the 'Vehicle Structure'.

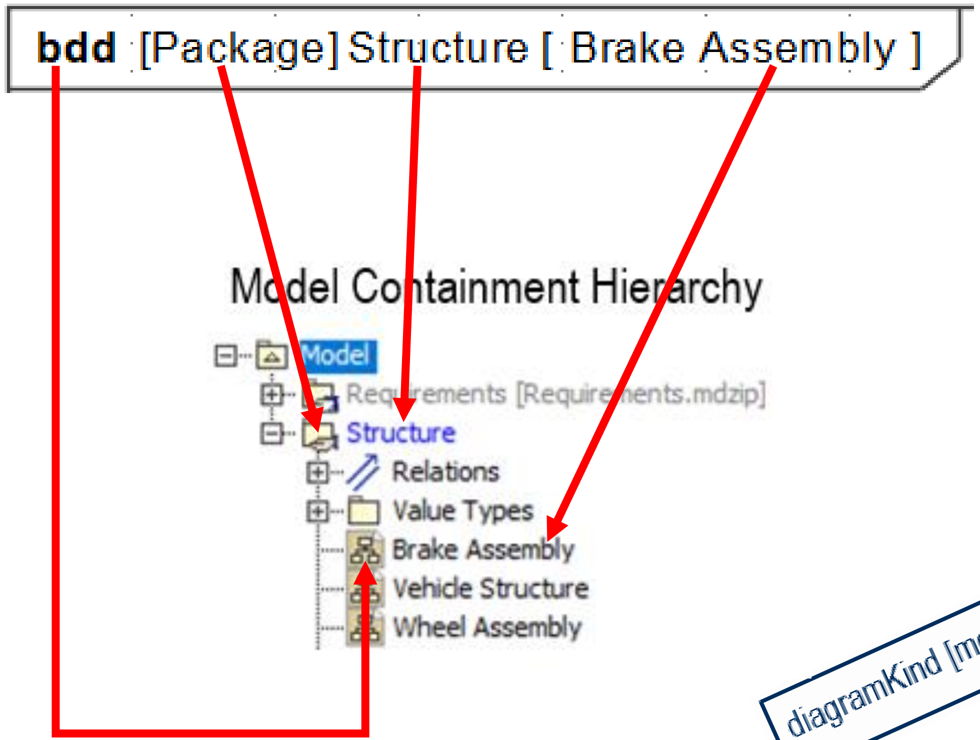

bdd [Package] Structure [Vehicle Structure]
Model Containment Hierarchy
Model
Requirements [Requirements.mdzip]
Structure
Relations
Value Types
Brake Assembly
Vehicle Structure
Wheel Assembly


diagramKind [modelElementType] modelElementName [diagramName]

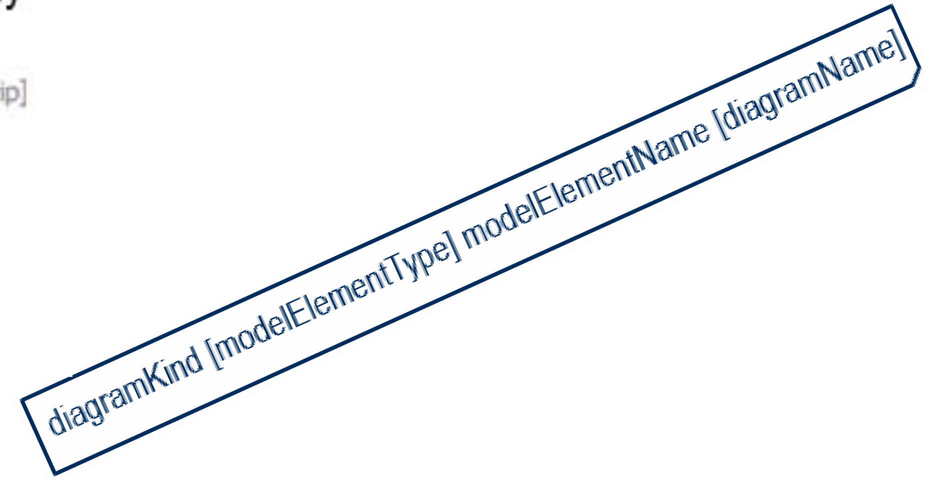
SysML Diagrams

Example Header #2

- This diagram is a 'bdd' of the 'Structure' 'Package' and it's purpose is to illustrate the 'Brake Assembly'.

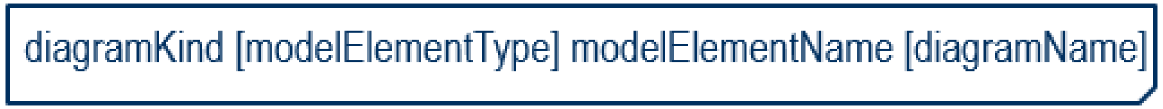


bdd [Package] Structure [Brake Assembly]
Model Containment Hierarchy
Model
Requirements [Requirements.mdzip]
Structure
Relations
Value Types
Brake Assembly
Vehicle Structure
Wheel Assembly



diagramKind [modelElementType] modelElementName [diagramName]

Namespace


diagramKind [modelElementType] modelElementName [diagramName]

- Namespace
  - Otherwise known as 'where is something contained in the model'
  - A diagram's modelElementName and modelElementType identify the <u>default</u> namespace for the elements shown on that diagram
  - A qualified name for a model element is required anytime the element shown on a diagram that is not contained within the default namespace
    - Why: to properly define the location of that element within the model
    - More details to follow during Package Diagram review

  - The question of Namespace can be asked 4 different ways, but they all essentially mean the same thing.
    - Assume an element named "CPU". One could ask:
      - What is the namespace for CPU?
      - Which element contains the CPU?
      - Which element owns the CPU?
      - Where is the CPU nested in the model hierarchy?

Elements of Definition vs. Elements of Usage

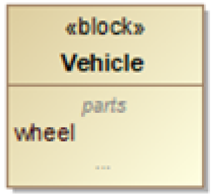
Concepts:

Elements have names, names can be anything, names imply nothing!!

An element must be <u>defined</u> in your model before it can be <u>used</u> in your model!

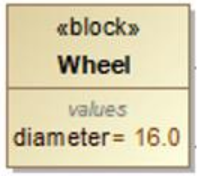
Example:

- It is not enough to simply state that you will have a thing named ‘wheel’ as a part of a vehicle.
- Just because we named the thing a ‘wheel’ does not automatically make it a commonly understood vehicle wheel !!

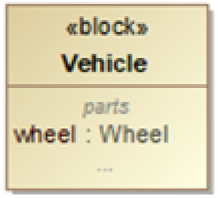


Incomplete

- You must define the Wheel



- Then you can identify the part of the Vehicle that you named ‘wheel’ as a type of ‘Wheel’

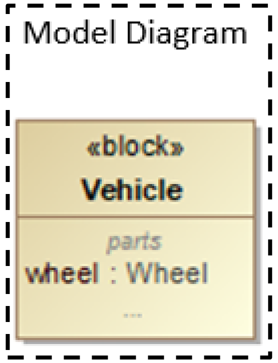


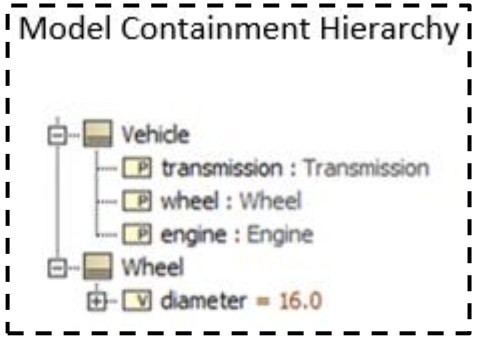
Complete

Elements of Definition vs. Elements of Usage

- Elements of Definition
  - Define ‘types’ of things that could exist within your system
  - Have a ‘name’ only

- Elements of Usage
  - Represent usages of those ‘types’ of Elements of Definition
  - Have a ‘name’ and a ‘type’ separated by a colon (name : type)
  - The ‘type’ will be the name of an element of definition somewhere within your model




Model Containment Hierarchy

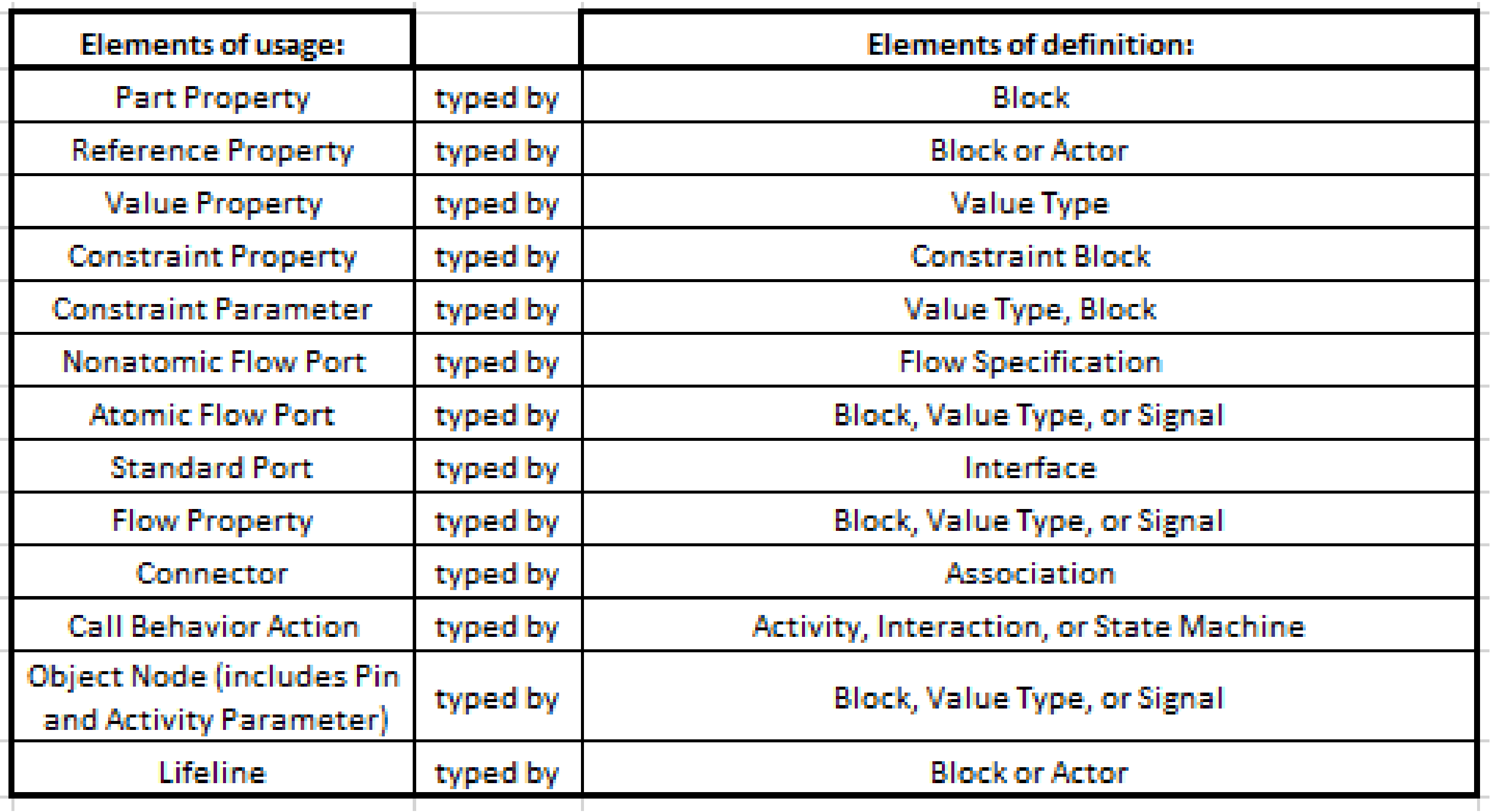
```
Vehicle
  transmission : Transmission
  wheel : Wheel
  engine : Engine
Wheel
  diameter = 16.0
```

Elements of Usage

The table provides a summary of:

Which kinds of elements can be elements of usage

Which kinds of elements of definition that the elements of usage are allowed to be ‘typed by’



Elements of usage:		Elements of definition:
Part Property	typed by	Block
Reference Property	typed by	Block or Actor
Value Property	typed by	Value Type
Constraint Property	typed by	Constraint Block
Constraint Parameter	typed by	Value Type, Block
Nonatomic Flow Port	typed by	Flow Specification
Atomic Flow Port	typed by	Block, Value Type, or Signal
Standard Port	typed by	Interface
Flow Property	typed by	Block, Value Type, or Signal
Connector	typed by	Association
Call Behavior Action	typed by	Activity, Interaction, or State Machine
Object Node (includes Pin and Activity Parameter)	typed by	Block, Value Type, or Signal
Lifeline	typed by	Block or Actor

VERY IMPORTANT TO REMEMBER THESE RELATIONSHIPS

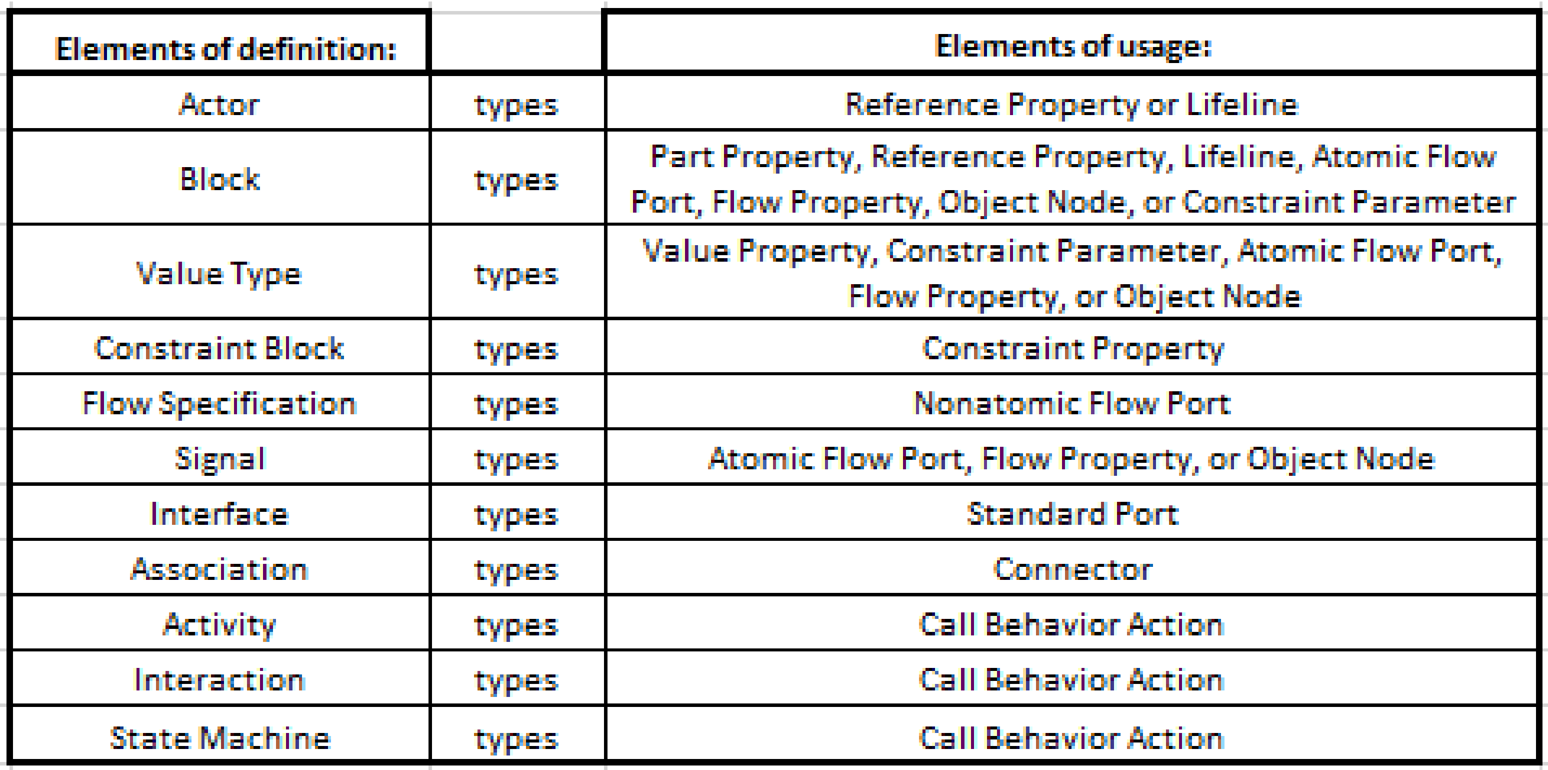
COLLEGE of CHARLESTON

Elements of Definition

The table provides a summary of:

Which kinds elements can be elements of definition

Which kinds of elements of usage that the elements of definition are allowed to 'type'



Elements of definition:		Elements of usage:
Actor	types	Reference Property or Lifeline
Block	types	Part Property, Reference Property, Lifeline, Atomic Flow Port, Flow Property, Object Node, or Constraint Parameter
Value Type	types	Value Property, Constraint Parameter, Atomic Flow Port, Flow Property, or Object Node
Constraint Block	types	Constraint Property
Flow Specification	types	Nonatomic Flow Port
Signal	types	Atomic Flow Port, Flow Property, or Object Node
Interface	types	Standard Port
Association	types	Connector
Activity	types	Call Behavior Action
Interaction	types	Call Behavior Action
State Machine	types	Call Behavior Action

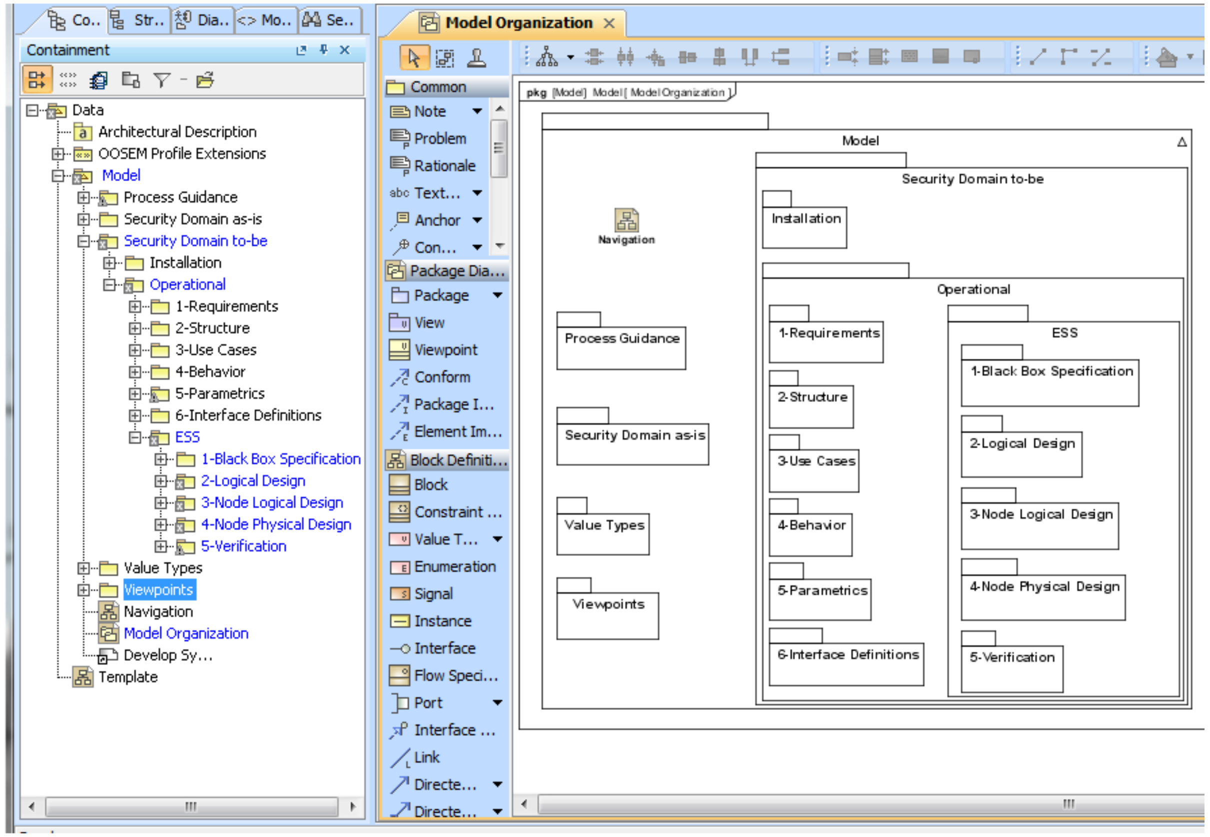
VERY IMPORTANT TO REMEMBER THESE RELATIONSHIPS

Model Conventions and Standards

- Ensure consistency across the modelDefined naming conventions for each type of model element (i.e. package, block, activity) and diagram name
- Defined templates for diagram types
- Defined stereotypes

Model Organization

- Package Structure
  - Top-level packages for ‘as-is’ and ‘to-be’ to capture current system and desired system, respectively
  - Lower-level packages contain:Packages that mirror the system hierarchy



© 2011 Elsevier, Inc.: A Practical Guide to SysML

Questions

COLLEGE of CHARLESTON

Summary

- Naming conventions, templates, and stereotypes are defined to ensure consistency across the model
- Model complexity requires that a well thought-out package structure be developed in order to maintain configuration control of the model

COLLEGE of CHARLESTON

References

Additional information can be obtained by reviewing:

SysML Distilled (Delligatti)

Chapter 2: Overview of the Systems Modeling Language