Common SysML Conceptual Stumbling Blocks

San Diego INCOSE Mini-Conference 2013.11.02
Rick Steiner
fsteiner@raytheon.com
rick@ricksteiner.net

Portions of this work are from the book *Practical Guide to the Systems Modeling Language (SysML)*, by Sanford Friedenthal, Alan Moore, and Rick Steiner, published by Morgan Kaufmann Publishers, Copyright 2008, 2012 Elsevier Inc. All rights reserved.
What to Systems Engineers have problems with when learning SysML?

1. 

   *SysML is too complex!* (flexibility vs. complexity)

2. 

   *What does that darn diagram header mean?* (models vs. diagrams)

3. 

   Why do I need both ibds and bdds? (definition vs. use)

4. 

   *Activity diagrams have no activities on them!* (activity modeling vs. functional hierarchy)

5. 

   *Why not use packages for my product breakdown structure?* (composition vs. containment)

6. 

   How is SysML different than Matlab? (parametric modeling)

7. 

   Why do I care about Units? (values, value types, units and quantity kinds/dimensions)

8. 

   Why isn’t SysML executable? (functionality vs. model execution)

* Recommended to be discussed at the SDINCOSE mini-conference
Issue #1: SysML is too complex!  What do you need?

“Traditional” SE can be done with only 3 Basic Diagram Types
Motivation for the Systems Modeling Language

- **Systems Engineers needed a standard graphical modeling language**
  - Many different modeling techniques/languages have been used for developing system models:
    - Behavior diagrams, IDEF0, N2 charts, Hatley-Pirbhai
    - Tools tended to support only one of these techniques/languages
  - The SE discipline has lacked a broad-based standard that to support general purpose system modeling needs

- **Why use the Unified Modeling Language (UML) as a basis?**
  - UML has become a de facto standard within software engineering
  - UML tools and training are widely available, and generally adaptable
  - OMG standardization process supports UML customization for specific domains (e.g., systems engineering, real-time, SOA, etc.)

- **SysML was developed to meet broad SE application**
  - SE Domain Special Interest Group jointly sponsored by INCOSE and OMG in 2001
    - Submissions needed to demonstrate their compliance with those requirements.
    - A panel of INCOSE experts evaluated this compliance before adopting SysML 1.0 in 2006.
  - Adoption of SysML by industry indicates that it has met the objective of being a broadly useful system modeling language.
Issue #2: Those darned Diagram Headers! (Models vs. Diagrams)

- Powerpoint & Visio teach us to take diagrams at face value
  - They can be manipulated independently, easily abstracted
  - I can add whatever I want, without rigor or process
- Good modeling tools tend to reinforce this impression
  - Drag and drop symbols, easy relationships
- But good modelers work mostly from the model browser
  - Drag elements from the browser on to the diagram
  - Can actually build models without using a diagram at all!
  - Tools render/update diagrams from the model
- Remember that reports, scripts and checkers run on the Model, not the Diagrams!
- Tables and matrices are also excellent model viewers
- Good modeling practices are consistent with good diagrams
  - Minimize number of elements at each layer of abstraction
  - Elide unnecessary information on the diagram… use views/viewpoints
  - Give the diagram a meaningful name!
SysML Diagrams exist in Model Context

- Each SysML diagram represents a model element
  - This provides the necessary context for every diagram
- Each SysML Diagram must have a Diagram Frame
- Diagram context (the element it represents) is indicated in the header:
  - Diagram kind (act, bdd, ibd, seq, etc.)
  - Model element type* (activity, block, interaction, etc.)
  - Model element name
  - Descriptive diagram name*

*Some parts of the diagram header can be elided, if it doesn’t cause confusion

```plaintext
diagramKind [ModelElementType] ModelElementName [User Defined Diagram Name]
```
Issue #3: *Definition* and *Use* in SysML

- This is the biggest “stumbling block” for new SysML users without a software background
  - “Why do I need BOTH a **bdd** and an **ibd**?”
  - If you need to ask the question, just build the **ibd**!
    - **bdd’s** are NOT mandatory, but you will eventually want them.
    - Building the bdd later will cause minor rework.
    - **bdd’s** and **ibd’s** are complimentary aspects of a single structural model
  - Same principle applies to parametric & activity models

- **Definition vs. Use:**
  - **Define** a model element in one place in the model.
  - **Use** it everywhere it is needed. Don’t redefine it unless you need to.

- See the “structure” corner of the pillars chart

- Mechanical engineers have less problem with this
  - They are used to defining parts, then using them
Example of Definition vs. Use

**Definition**

- **ibd** [Block] Anti-Lock Controller [Basic]
  - d1 : Traction Detector
  - c2 :
  - m1 : Brake Modulator

**Use**

- **bdd** [Package] Structure [ABS Structure Hierarchy]
  - <<block>> Library:: Electronic Processor
  - <<block>> Anti-Lock Controller
  - <<block>> Library:: Electro-Hydraulic Valve
    - d1
    - m1
  - <<block>> Traction Detector
  - <<block>> Brake Modulator
Issue #3a: Activity Modeling & Functional Hierarchy

- Activity models are intuitive
  - Functional flow, control & data flow…

- Activity diagrams represent **Use**
  - What about **Definition**? The bubbles on act’s are Actions, not Activities!

- Activity models are usually built without regard to functional hierarchy
  - Tools don’t support functional hierarchy very well.

- Recommendation:
  - Don’t put an Action on an activity diagram without having it call an Activity. Use only CallBehaviorActions.
  - Manage the Activities in a separate set of packages, just like blocks.
  - Generate functional hierarchies using **bdd’s**
Activity Model & Functional Hierarchy

FIGURE 8.1
An example activity diagram.

FIGURE 8.2
An example of an activity hierarchy in a block definition diagram.
Issue #4: Containment vs. Composition

- This is a common issue for even experienced modelers
- Containment relation is seldom seen on diagrams (crosshair), but is the basis for the model browser.
  - Packages exist as “containers” for organizing the model
  - Blocks, Requirements can also be “containers”… but I advise against it!

- Composition (black diamond) is a way of representing Use on a Definition diagram.
  - Composition role names on the bdd correspond to part/action names on the ibd/act.
Containment in Browser & Package Diagram

FIGURE 5.3
Showing nested packages on a package diagram.

FIGURE 5.4
Browser view of the model’s package hierarchy.
Containment/Package Structure
Suggestions

- Structural hierarchy:
  - Treat every block as reusable
    - Put them in a package where modelers can find them… based on what they are, not how they are used
  - Build ibd’s from blocks in the browser (drag & drop). Autorender the bdd’s after the ibd’s have been built.
    - Untyped parts are poor modeling form
  - Enterprise Architect Demo

- Functional hierarchy
  - Treat every activity as reusable
  - Build act’s from activities in the browser. Most tools won’t autorender bdd’s, so this will be a little more work.

- Data/Info Model
  - Enforce all data types in one convenient containment hierarchy
  - Put FlowSpecifications or InterfaceBlocks in the hierarchy too
Package Diagram
Organizing the Model

By Diagram/Model Type

- Use Cases
- Requirements
- Behavior
- Structure
- EngrAnalysis

By Abstraction Level

- Enterprise
- System
- Logical Design
- Physical Design
- Verification

By IPT

- Architecture Team
- Requirements Team
- IPT A
- IPT B
- IPT C
Issue #5: SysML vs. Matlab (Descriptive vs. Analytical Models)

- SysML models are descriptive
- Performance models (like Matlab) are analytical
Descriptive Models & Analysis Models

Model Based Engineering Discipline

External Requirements

System Documentation & Specifications

Descriptive/Specification Model

Analysis Models & Simulations

Linkage to Other Disciplines/IPTs

Arrows indicate interactions that could potentially be automated

traceability, rationale

view-point

structure

behavior

requirements

parametrics

analysis needs

performance estimates

closed form

discrete event

network

analysis models & simulations
Parametric Modeling Links Descriptive & Analytical Models

- Systems engineers tend not to build parametric models
  - Powerful, unique capability to check model consistency and integrity
  - Linkage to external analysis capabilities/simulations/models
  - Test and validation implications

- This is changing with better vendor support
  - Phoenix ModelCenter “MBSE Pak” plugin
  - InterCAX plugin for MagicDraw and Rhapsody

- When this capability works, it is more impressive, meaningful, and powerful than animating the SysML model!
Parametric Example: Air Compressor – Definition & Use

Figure 3.11

Figure 3.12
We don’t think about adding specific values until the performance analysis has already started. Once we start adding values, we discover that we need units and dimensions too. This adds considerable work for the modeler.

Default values can be added to the block, but actual values require instance specifications. Modelers tend to have initial difficulty with instances. Most tools don’t support this well.

Standard units and dimensions can be imported as library. But custom units and dimensions are almost always required for a new domain. ValueTypes must be built from the Units and Dimensions before they can be used. Not a big job, but often overlooked.

“A model without values has no value” – Dr. Darren Kelly
Issue #7: Modeling Functionality vs. Model Execution

- Model execution is becoming important for checking dynamic consistency of the system design.
- “SysML doesn’t execute”
  - It was never intended to! Nor does it calculate.
  - Some tools provide execution capability, if you follow their modeling rules (Rhpsody, MagicDraw Cameo, etc.)
- SysML is compatible with foundational UML (fUML) and its action language (ALF)
- The burden of model execution: In addition to a semantically correct behavioral model, there needs to be:
  - A simulation environment, including keeping track of model time
  - A stimulus or input file
  - A visualization/animation capability
  - Resource constraint/utilization based on allocation to structure
- Some of the value of model execution may be gained using linked external models, e.g. Extend
- System model execution is not a substitute for MS&A.
- Choose, but choose wisely!
4 Pillars of SysML – ABS Example

1. Structure

2. Behavior

3. Requirements

4. Parametrics
Cross Connecting Model Elements in ABS example

1. Structure

Vehicle System Specification

- Vehicle Specifications
  - [package] VehicleSpecifications
    - [Requirements Diagram - Braking Requirements]
      - req [package] VehicleSpecifications
      - Vehicle System Specification
      - Vehicle Subsystem Specification
      - [requirement] Stopping Distance
        - id = 102
        - text: The vehicle shall stop from 60 mph within 150 ft on a clean dry surface.
        - Verify [interaction] Minimum Stopping Distance
        - SatisfiedBy [block] Anti-LockController
      - [requirement] Anti-Lock Performance
        - id = 337
        - text: Braking subsystem shall prevent wheel lockup under all braking conditions.
        - SatisfiedBy [block] Anti-LockController

2. Behavior

- Structure
  - ibd [block] Anti-LockController [Internal Block Diagram]
  - satisfies [requirement] Anti-Lock Performance
    - allocatedFrom [section] d1: Traction Detector
    - d1: Traction Detector
    - allocatedFrom [section] m1: Brake Modulator
    - m1: Brake Modulator

- Behavior
  - a1: Detectors Of Traction
  - a2: Modulate Braking Force

3. Requirements

- Verify [interaction] Minimum Stopping Distance

4. Parametrics

- v.b.abs.m1.duty cycle : %
- v.c.friction : N
- v.b.r.braking force : N
- v.position : m
- clk.time : sec

Allocate [ObjectFlow] objFlow1

Allocate [block] Anti-LockController

Allocate [block] Traction

Allocate [block] Brake Modulator

Value [Element] Duty Cycle: Percentage

Begins with:

- Structure

Connections:

- satisfy [requirement]
- allocate [ObjectFlow]
- value binding [requirement]
- deriveReqt [cross connecting]

Ends with:

- Verify [interaction] (via interaction)
- Parametrics

- Raytheon
"SysML Light" (Basic Capability)
Enhancing Education Using Constraint Graph-based Knowledge Representations [Cowan et al.]

Initial results with high school physics class: 
*Students using constraint graphs did 70% better*

Figure 7. Free body diagram and constraint network with addition of $\tau$

“\textit{I believe that this process will be helpful to others because I have been doing the same thing in my head to organize and understand the different equations and to help me solve the problems successfully.}”

[student comment]

Three Ways to Model Behavior:
Each Invokes Activities

State X
Activity X1
Transition Activity T1

State Y
Activity Y1
Activity Y2

Activity Model

Class 2
Op 2.1 (msg:type)

Class 1
Op 2.1

Interaction Model
(sequence diagram)