Introduction to Model-Based Engineering

What does a good model smell like?

Dr. Bruce Powel Douglass, Ph. D.
Principal
A Priori Systems
www.bruce-douglass.com
About the Author

Bruce Douglass, Ph.D.

- Senior Principal Agile Systems Engineer
- Systems Engineering Tech Center
- The MITRE Corporation
- Can be reached at bdouglass@mitre.org
- Contributor to UML standard
- Contributor to SysML standard
- Developer of UML Dependability Profile
- Former Cochair RTAD Task Force for the OMG
A copy of *Agile Model-Based Systems Engineering Cookbook* will be given away at the end of the session. *You must be present to win.* If you do not acknowledge your presence when called, another attendee will be selected.
Starting Definitions

• **Model**
  – is a representation of a system of interest from a particular viewpoint, capturing attributes for a specific purpose. A model is always an abstraction in that it focuses on properties of interest at the expense of properties not of interest and at a specified level of precision (detail).

• **MBE (Model-Based Engineering)**
  – “An approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle.” (Final Report, Model-Based Engineering Subcommittee, NDIA, Feb. 2011)

• **MBSE (Model-Based Systems Engineering)**
  – “The formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” (INCOSE MBSE Report, September 2007)

• **MDD (Model-Driven Development)**
  – The use of models for the specification and design of software-based systems.
Starting Definitions

• DE (Digital Engineering)
  – Digital engineering is the ability to perform discipline-specific engineering by collaborating with other disciplines and by leveraging authoritative system data in digital form from those disciplines within my tools of choice and in the right format.

• DE Platform
  – A standard platform for projects which includes pre-installed tools, tool integrations, processes, and links to training and other knowledge / skill resources with the intent of allows quick start up of internal and sponsor-related projects.

• Digital Thread
  – A connected set of models of a system in different lifecycle stages, including specification, design, operation, and maintenance.

• Single Source of Truth
  – Each important datum is located in a singular, authoritative place and is connected, via navigable links, to all other relevant data in that or other repositories. Note: this doesn’t mean that all data are in the same repository but the authoritative source for each datum is singular.
Modeling For Beginners

Drawing vs Modeling
What’s a model?
Models & Views
Foundational Concept of Modeling

- A drawing is a picture with only imagined semantics and no underlying repository of information.
- A model focuses on system aspects of interest and ignores others.
- Stores underlying semantics in model repository.
- Generates any needed documentation from the model repository.
- Uses a precise language.
- Supports verification through review, execution and/or formal methods.

Once you’re done drawing, then go do the “real work”.

Note: it IS possible to use a modeling tool solely for drawing and not modeling, but it’s not a good idea!
So What IS a Model exactly?

Modeling is the development of a set of system data of relevant systems and their properties

Models have views (e.g. diagrams)

Diagrams show subsets of eng. data

Diagrams have singular purpose

Diagrams answer questions

Diagrams support specific reasoning

Models have scope

Models have purpose

Models have accuracy

Models have fidelity

Models are falsifiable

Models are verifiable

Models are interconnected data!
To be clear, you do NOT model in Visio or PowerPoint; you can only draw pictures.
Uses of Diagrams and Tabular Views

• Data Entry
  – Drawing diagrams or entering data into tables/matrices is a way of entering information into the model
  – When you create an element on the diagram, the model either
    o Refers to an existing element, and updates it based on your actions, or
    o Creates a new element in the model repository

• Model visualization
  – Creating a diagram or tables allows you to create a view of a subset of the model information

• Simulation / Execution Debugging & Execution Control
  – Some modeling tools provide special diagrams and tools to control execution, insert events, change values, set breakpoints, etc.
Executable Models

• WHY
  – To make sure the model isn’t stating “utter nonsense.”
  – Models make declarative and imperative statements of truth
  – It is absolutely crucial that we have a means by which we can verify that the statements of truth made by the model can be verified or demonstrated to be true
    o Such models are said to be “falsifiable”; this means that there is a way to demonstrate that a false model is indeed false
  – The larger the model, the more important this is
  – The more significant the impact of the model or system, the more important this is

• Rhapsody, Magic Draw, and Sparx Enterprise Architecture can build and execute models (with differing levels of fidelity)

“Any language rich enough to say something interesting is also rich enough to say utter nonsense that at first glance sounds reasonable.”
- Douglass’ Paradox

Declarative statements identify what you want to happen; imperative statements identify how to make something happen.
What do we mean by “verification & validation” of work products (e.g. models)?

**Syntactic: Is it well-formed?**

- “Compliance in form”
  - Performed by quality assurance personnel
  - **Audits** – work tasks are performed as per plan and guidelines
  - **Syntactic review** – work products conform to standard for organization, structure and format

**Semantic: Is the content correct?**

- **Compliance in meaning**
  - Performed by engineering personnel
  - Three basic techniques
    - **Semantic review** (subject matter expert & peer) – most common, weakest means
    - **Testing** – requires executability of work products, impossible to fully verify
    - **Formal methods** – strongest but hard to do and subject to invariant violation

**Valid: Does it solve the right problem?**

- **Validation** = “meets the stakeholder need”
  - Performed by customer + engineering
  - Some common techniques
    - **Review** – (subject matter expert & customer) – most common, weakest
    - **Simulation** – show simulated input → outputs
    - **Sandbox** – exploratory usage in constrained environment
    - **Flight test** – demonstration of system capabilities
    - **Deployment** – early usage of system of partial capability
# INCOSE Organizational Model-Based Capabilities Matrix

<table>
<thead>
<tr>
<th>Level</th>
<th>Benefit</th>
<th>Focus</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>High</td>
<td>Employing modeling as an organizational standard approach; managed reusable DE Assets</td>
<td>DE Platform infrastructure, tooling, training, and processes widely applied across organization</td>
</tr>
<tr>
<td>3</td>
<td>Moderately High</td>
<td>Wide-scale use of modeling throughout projects</td>
<td>Model integrated with other functional disciplines, digital threads defined and digital twin</td>
</tr>
<tr>
<td>2</td>
<td>Moderate</td>
<td>Standardizing use of modeling</td>
<td>Integration of modeling into processes, standardized reviews and quality assurance</td>
</tr>
<tr>
<td>1</td>
<td>Low</td>
<td>Answer specific questions during development</td>
<td>Modeling efforts address specific objectives and questions</td>
</tr>
<tr>
<td>0</td>
<td>None</td>
<td>Little or no use of models in systems engineering efforts; use of document-based, siloed data.</td>
<td></td>
</tr>
</tbody>
</table>

For more information, visit: https://connect.incose.org/Pages/Product-Details.aspx?ProductCode=MBCM
# Project-Oriented Modeling Maturity

<table>
<thead>
<tr>
<th>Level</th>
<th>Benefit</th>
<th>Focus</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Integrated cross platform modeling</td>
<td>High</td>
<td>Large-scale breaking down with federated models; connecting tools and data</td>
</tr>
<tr>
<td>3</td>
<td>Executable</td>
<td>Moderately High</td>
<td>Use of verifiable, testable models; Executive state and activity models, model-based test; use of quantitative metrics</td>
</tr>
<tr>
<td>2</td>
<td>Standardization</td>
<td>Moderate</td>
<td>Wide-spread use of modeling within the project; single source of truth</td>
</tr>
<tr>
<td>1</td>
<td>Visualization</td>
<td>Low</td>
<td>Visualizing engineering data</td>
</tr>
<tr>
<td>0</td>
<td>Textual / code-based/ siloed document-based development</td>
<td>None</td>
<td>Manual, time intensive heroic development with disconnected, siloed data</td>
</tr>
</tbody>
</table>
At the UML 101/SysML 101 level, they are the same, except some elements are renamed.
Definition: An architecture framework is an encapsulation of a minimum set of practices and requirements for artifacts that describe a system's architecture.
What is UML?

Unified Modeling Language
Diagrams and views
Model elements
4-Tier Metamodel Architecture
What is UML?

- Comprehensive full life-cycle 3rd Generation modeling language
  - Standardized in 1997 by the Object Management Group (OMG)
  - Incorporates state of the art Software and Systems development concepts
- Matches the growing complexity of real-time systems
  - Large scale systems, Networking, Web enabling, Data management
- Extensible and configurable
- UML supports but doesn’t require object-oriented development
- UML is process agnostic
  - By design, the UML is meant to be used with any reasonable development process
UML Features

• UML is a graphical language
  
  – **Diagrams** form the primary means by which models are created and understood
  
  – **Packages** are folders that contain model elements including both diagrams and the elements they portray.
    
      o This applies to the UML itself but also to the user models (designs) you create
  
  – The key is the underlying semantic repository of information about the system you’re modeling
  
  – A **diagram** type is defined by the types of things that can be represented and their symbology
  
  – A **diagram usage** is the purpose for a diagram, which subsets the kinds of elements used
  
  – Example:
    
      o A class diagram is a type of UML diagram
      
      o Uses of class diagrams: class, structure, object, package, task, subsystem, architecture, interface
UML Semantic basis

- UML is constructed using a 4-tier metamodel hierarchy
  - M3 – Meta-metamodel (MOF Core language)
  - M2 – Metamodel (UML Language)
  - M1 – Design model (model)
  - M0 – Instance model (deployed system)
- The UML definition itself is divided up into packages to support
  - Modularity
  - Layering
  - Partitioning
  - Extensibility
  - Reusability

“It’s Meta-Turtles all the way down”
UML Diagrams

From UML 2.51 OMG Document Number formal/2017-12-05
What is SysML?

SysML is derived from UML
SysML Timeline
UML vs SysML
What is SysML?

• A graphical modeling language in response to the UML for Systems Engineering RFP developed by the Object Management Group (OMG), International Council on Systems Engineering (INCOSE), and AP233
  – a UML Profile that is both a subset and extension to UML 2
• Designed specifically for the Systems Engineering domain with extensions for requirements and analysis
• Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
• SysML is the most common way to represent systems engineering information in a rigorous, structured way by storing the information in models. We discuss models in more detail shortly.
• The pervasive application of models for systems engineering is known as Model-Based Systems Engineering (MBSE)

Important! At a basic level of use, UML and SysML are the same language, with only minor naming differences between them.
  – More advanced uses of SysML will highlight the differences between them.

Like UML, SysML is a language and is process-agnostic.
Nine SysML Views

The nine SysML diagrams are categorized as follows:
– Behavioral Diagrams - dynamic change of system **behavior** over time
– Structural Diagrams - static system **structure** diagrams
– Requirements Diagram

SysML Diagrams

Behavorial

- Use Case Diagram (uc)
- State Machine Diagram (stm)
- Sequence Diagram (sd)
- Activity Diagram (act)

Requirements (req)

- Allocation Table
- Parametric Diagram (par)

Structural

- Internal Block Diagram (ibd)
- Package Diagram (pkg)

- Block Definition Diagram (bdd)

Source: OMG Systems Modeling Language v1.6, page 211
<table>
<thead>
<tr>
<th>View</th>
<th>Type</th>
<th>UML2 Analog</th>
<th>Lifecycle usage</th>
<th>Essential</th>
<th>Dynamic simulation</th>
<th>Computational</th>
<th>Supports code gen</th>
<th>Formal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Diagram (req)</td>
<td>Static Functionality</td>
<td>n/a</td>
<td>Requirements Specification; Functional Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use Case Diagram (uc)</td>
<td>Static Functionality</td>
<td>Use case diagram</td>
<td>Requirements Specification; Functional Analysis</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity Diagram (act)</td>
<td>Dynamic Behavior</td>
<td>Activity diagram – minor changes</td>
<td>All</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Sequence Diagram (sd)</td>
<td>Interaction Behavior</td>
<td>Sequence Diagram</td>
<td>All</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Diagram (stm)</td>
<td>Dynamic Behavior</td>
<td>State Diagram</td>
<td>All</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Definition Diagram (bdd)</td>
<td>Static Structure</td>
<td>Class Diagram (moderate change)</td>
<td>Architecture; Design</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Block Diagram (ibd)</td>
<td>Static Structure</td>
<td>Structure Diagram (moderate change)</td>
<td>Architecture; Design</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parametric Diagram (par)</td>
<td>Static Functionality</td>
<td>n/a</td>
<td>All</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package Diagram (pkg)</td>
<td>Static Structure</td>
<td>Package diagram</td>
<td>All</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Table</td>
<td>Static Table</td>
<td>n/a</td>
<td>Requirements Specification; Functional Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allocation Matrix</td>
<td>Static Matrix</td>
<td>n/a</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Learning SysML: The A Priori Curriculum
Example

A quick look at the Pegasus Smart Bike Trainer
Adapted to Cameo Magic Draw from the Rhapsody models within the book
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DlReg01</td>
<td>In DI Shifting mode, if the UP button is pressed and the system is already in the highest possible gear, then the system shall audibly beep and keep the current gearing.</td>
</tr>
<tr>
<td>2</td>
<td>DlReg02</td>
<td>The system shall provide shifting with a DI Shifting mode that enables the DI shifting buttons and disables the shifting levers.</td>
</tr>
<tr>
<td>3</td>
<td>DlReg03</td>
<td>In DI Shifting mode, the UP button shall shift into the next highest possible gearing from the selected gear set, as measured in gear inches.</td>
</tr>
<tr>
<td>4</td>
<td>DlReg04</td>
<td>In DI Shifting mode, the UP button shall shift into the next highest possible gearing from the selected gear set, as measured in gear inches.</td>
</tr>
<tr>
<td>5</td>
<td>DlReg05</td>
<td>In DI Shifting mode, if the DOWN button is pressed and the system is already in the lowest possible gear, then the system shall audibly beep and keep the current gearing.</td>
</tr>
<tr>
<td>6</td>
<td>DlReg06</td>
<td>In DI Shifting mode, when an upshift requires changing the chain ring, the system shall progress to the next largest gearing, as measured by gear inches.</td>
</tr>
<tr>
<td>7</td>
<td>DlReg07</td>
<td>In DI Shifting mode, when a downshift requires changing the chain ring, the system shall progress to the next smallest gearing, as measured by gear inches.</td>
</tr>
<tr>
<td>8</td>
<td>DlReq10</td>
<td>The system shall enter DI Shifting Mode by selecting that option in the Configuration App.</td>
</tr>
<tr>
<td>9</td>
<td>DlReq11</td>
<td>Once DI Shifting mode is selected, this selection shall persist across resets, power resets, and software updates.</td>
</tr>
<tr>
<td>10</td>
<td>DlReq12</td>
<td>Mechanical shifting shall be the default on initial start up or after a factory-settings reset.</td>
</tr>
<tr>
<td>11</td>
<td>DlReq13</td>
<td>The system shall leave DI Shifting mode when the user selects the Mechanical Shifting option in the Configuration App.</td>
</tr>
<tr>
<td>12</td>
<td>efarg01</td>
<td>The system shall notify the rider of the current number of chain rings and cassette rings on start up.</td>
</tr>
<tr>
<td>13</td>
<td>efarg02</td>
<td>The system shall accept a rider command to enter a mode to configure the gearing.</td>
</tr>
<tr>
<td>14</td>
<td>efarg03</td>
<td>The system shall accept a rider command to set up from 1 to 3 front chain rings, inclusive.</td>
</tr>
<tr>
<td>15</td>
<td>efarg04</td>
<td>The default number of chain rings shall be 2.</td>
</tr>
<tr>
<td>16</td>
<td>efarg05</td>
<td>The rider shall be able to decrement the cassette ring from a higher (smaller number of teeth) to the next lower (larger number of teeth) gear until the largest cassette ring is reached.</td>
</tr>
<tr>
<td>17</td>
<td>efarg06</td>
<td>The system shall accept a rider command to set up from 10-12 cassette rings, inclusive.</td>
</tr>
<tr>
<td>18</td>
<td>efarg07</td>
<td>The default number of cassette rings shall be 12.</td>
</tr>
<tr>
<td>19</td>
<td>efarg08</td>
<td>The system shall accept a rider command to set any chain ring to have from 20 to 70 teeth.</td>
</tr>
<tr>
<td>20</td>
<td>efarg09</td>
<td>The system shall accept a rider command to set up any cassette ring to have from 10 to 50.</td>
</tr>
<tr>
<td>21</td>
<td>efarg10</td>
<td>The default number of teeth for 1 chain ring shall be 48.</td>
</tr>
<tr>
<td>22</td>
<td>efarg11</td>
<td>The default number of teeth for 2 chain rings shall be 34 and 53.</td>
</tr>
</tbody>
</table>
Use Case Diagram
Type Context
Type Composition Architecture
Connected Architecture
An interaction

reqSetPedalPosition("pos = 3.14156")
reqSetPedalSpeed("pspeed = 90")
reqSetMeasuredPedalForce("f = 206.181")

tm("60")

tm("5000")
reqSetPedalPosition("pos = 3.76587")
reqSetPedalSpeed("pspeed = 90")
reqSetMeasuredPedalForce("f = 193.82")

calculateInertia()
Flow of control behavior
Some state behavior