Introduction to System Modeling and Ontologies

Steven Jenkins
Engineering Development Office
Systems and Software Division
Motivating example
Reasoning about models
Some objectives of modeling
Presentations and facts
Ontologies and facts
JPL/IMCE ontologies
Ontology standards
Ontologies and SysML
Closing thoughts
Is This A Model?

- spacecraft
- transmit telemetry
- telemetry packet
- receive telemetry
- ground system

Sure, why not?
Is It A Good Model?

Not so much.
What’s Wrong With It?

- spacecraft
- transmit telemetry
- telemetry packet
- receive telemetry
- ground system

Same symbol for different kinds of things
Same symbol for different kinds of relationships
Better?

Not much. Essential distinctions are merely suggested--they should be explicit.
Rather than merely hinting at distinctions with shapes or colors, we could devise a set of types or classes to be applied to model elements.

The set of types is application-dependent:
- Systems engineers talk about different things from chefs.
- The distinctions are whatever matters for your application.
- Is red wine a different type from white, or is it merely a property of wine?
  - It depends on what you want to say about wine.

What kinds of things do systems engineers talk about?
- Component, Interface, Function, Requirement, Work Package, Product, Process, Objective, Message, etc.

Let’s apply some classes to our model.

For now, every element has:
- one type, denoted like this: «type»
- one name, which identifies an individual of that type.
Model With Typed Elements

- «component» spacecraft
  - «function» transmit telemetry
  - «message» telemetry packet
- «component» ground system
  - «function» receive telemetry

Much better.
What components are present?

- **component**: spacecraft
  - function: transmit telemetry
  - message: telemetry packet
  - function: receive telemetry

- **component**: ground system
What functions are present?

- «component» spacecraft
  - «function» transmit telemetry
  - «message» telemetry packet

- «component» ground system
  - «function» receive telemetry
What messages are present?

- «message» telemetry packet
- «function» transmit telemetry
- «component» spacecraft
- «function» receive telemetry
- «component» ground system
Add Typed Relationships

Note that relationships are now directed.
What component performs the function *transmit telemetry*?
What functions does the component *ground system* perform?

- **spacecraft**
  - **transmit telemetry**
  - **message** telemetry packet
  - **receives** telemetry packet
- **ground system**
  - **receive telemetry**
  - **performs** telemetry packet

- **performs** telemetry packet
- **sends** telemetry packet
What messages does the function \textit{transmit telemetry} send?

- \textbf{«component» spacecraft}
- \textbf{«function» transmit telemetry}
- \textbf{«message» telemetry packet}
- \textbf{«performs»}

- \textbf{«sends»}

- \textbf{«component» ground system}
- \textbf{«function» receive telemetry}
- \textbf{«performs»}

- \textbf{«receives»}
What components perform a function that sends or receives the message *telemetry packet*?

Alternatively, what component designs may be affected if the definition of *telemetry packet* changes?
• We can use models to answer questions
• The questions may be about the system itself
  – What is it?
  – How does it work?
  – Is the performance adequate?
  – What happens if something breaks?
• The questions may be about the model
  – Is it complete?
  – Is it consistent?
  – Does it support required analyses?
• The questions may be about the design artifacts
  – Are all required documents present?
  – Does each document contain all required content?
• We call answering these kinds of questions reasoning
  – It doesn’t necessarily mean exotic, artificial intelligence
What components perform no function?

- spacecraft
- ground system

- transmit telemetry
- receive telemetry

- telemetry packet

«message»

«sends»

«receives»

«performs»
What functions are not performed by any component?
What messages are received but not sent?

«component» spacecraft
«function» transmit telemetry
«message» telemetry packet
«receives»

«function» receive telemetry
«component» ground system
«performs»
Reasoning About Completeness

What messages are sent but not received?

«component» spacecraft

«function» transmit telemetry

«message» telemetry packet

«sends»

«component» ground system

«function» receive telemetry

«performs»

«performs»
Reasoning About Consistency

Are there illegal or meaningless relationships in the model?

«component» 
spacecraft

«function» 
transmit telemetry

«message» 
telemetry packet

«function» 
receive telemetry

«component» 
ground system

«performs»

«sends»

«performs»

«sends»

«sends»

«receives»
Reasoning About Design

Rule: Reserve mass $m_r$ of any component with parts is the difference between its $m_a$ and the sum of $m_a$ of its parts.
Rule: Reserve mass $m_r$ of any component with parts is the difference between its $m_a$ and the sum of $m_a$ of its parts.

Rule: CBE mass $m_e$ of any component with parts is the sum of $m_e$ of its parts.

$m_e$: estimated mass
$m_a$: allocated mass
Reasoning About Design

Rule: Reserve mass $m_r$ of any component with parts is the difference between its $m_a$ and the sum of $m_a$ of its parts.

Rule: CBE mass $m_e$ of any component with parts is the sum of $m_e$ of its parts.

Policy: $m_e < m_a$ for every component.

$m_e$: estimated mass
$m_a$: allocated mass
Reasoning About Design

Is this OK?

Probably not. Requirements shouldn’t jump component levels.
Reasoning About Design

«component» spacecraft

«contains»

«component» telecom

«contains»

«component» propulsion

«requires»

«requirement» eirp

«refines»

«requirement» gain

«specifies»

«component» amplifier

«specifies»

«component» antenna

Is this OK?

Yes. This is a common pattern.
No. Requirement flowdown should be consistent with product decomposition.
Yes. Sometimes you decompose at the same level for clarity.
Some Objectives of Modeling

- To describe a design in durable form
  - You can use almost anything for that

- To communicate a design to a set of stakeholders
  - Now you need (at least) a common notation and familiar presentation idioms
  - Standards (e.g., SysML) cover most of that

- To organize and relate analyses of a design
  - This is, in general, a much harder problem
  - You have to make sure that every element that could affect an analysis is present, properly identified, and consistently related to appropriate other elements
    - This is largely outside the scope of SysML, except to provide extension mechanisms that allow you to define the rules
    - You also need software to reason about your models
      - This is also outside the scope of SysML, but some tools do
  - Analysis operates on facts
Presentations and Facts

Presentation

- «component» spacecraft
- «performs» transmit telemetry

Facts

- spacecraft is a «component»
- transmit telemetry is a «function»
- spacecraft «performs» transmit telemetry

SysML is (among other things) a presentation standard

We need other standards for our facts
Facts and Ontologies

- The field that deals with facts and reasoning is **logic**
- The subset of logic that deals with facts and their meaning is **ontology**
- Ontologies contain **axioms**:
  - Definitions of concepts and their specializations
    - e.g., a *Spacecraft* is a *Flight Component*, which is a *Component*
    - These are sometimes called **classes**
  - Definitions of attributes of individuals of a class
    - e.g., *mass* is a property of *Flight Component*
    - These are sometimes called **data properties**
  - Definitions of relationships among individuals
    - e.g., a *Component* performs a *Function*
    - These are sometimes called **object properties**
  - Restrictions
    - e.g., a *Function isPerformedBy* at most one *Component*
  - Facts about individuals using these concepts and properties
• There is a well-developed body of theory that can
  – help us avoid undecidable questions
    • i.e., not solvable in principle
  – help us avoid intractable questions
    • i.e., solvable in principle but not in practice

• There is a body of tools that can
  – help us edit our ontologies
  – validate our ontologies
    • i.e., tell us if they’re well-formed, consistent, and satisfiable
  – compute inferences
    • i.e., JEO is a Spacecraft and Spacecraft is a Component implies JEO is a Component
    • these are sometimes called entailments
  – answer a large class of questions about facts
    • i.e., What Components perform a Function that sends or receives the particular Message?
• **We use a lot of discipline-specific tools and terminology in space flight systems engineering**
  – e.g., trajectory synthesis, radiation effects modeling
  – *SysML* supports the broad discipline of systems engineering, but we need a unifying vocabulary that can relate these disciplines to each other
• **This problem is not unique to space flight (nor to systems engineering)**
  – Lots of people have been working on it for years.
• **There is a set of international (W3C) standards for defining and using ontologies**
  – All related to the Web Ontology Language (OWL)
• **We’re building OWL ontologies for disciplines of interest**
Foundation Ontologies
Base, Mission, Project, Physics, Quantities-Units-Dimensions-Values, Analysis, Artifact

Discipline Ontologies
- Mechanical
- Electrical
- Thermal
- Propulsion
- ACS, Nav, ...

Application Ontologies
Star Tracker, Sun Sensor, Reaction Wheel, Thruster, Antenna, ...

Fundamental terms use in all projects, disciplines, and applications

Discipline-specific terms specified and owned by cognizant organizations

Focus is integration and interoperation

Kinds of items that are modeled in a project

Focus is reuse
Concepts in OWL are typically capitalized, properties are not.
Foundation Ontology Example

- Item
  - Requirement: sends, receives
  - Function: specifies, performs
  - Flow: invokes, emits, ingests
  - Mission: performs
  - Component: performs
Foundation Ontology Example

- Component
  - Requirement
    - Function
      - Process
      - Product
      - authorizes
      - produces
      - supplies
  - authorizes
Foundation Ontology Example

- **RecDel**
  - receives
  - delivers
  - invokes

- **Requirement**
  - specifies

- **Process**
  - performs

- **Organization**
  - authorizes

- **Quantification**
  - quantifies

- **WorkPackage**
• **SysML contains an extension mechanism for user-defined types and properties**
• A collection of these extensions is called a *profile*
• We generate profiles by transforming ontologies
• This ensures that
  – OWL concept and property definitions are consistent with SysML stereotypes
  – SysML “instance” models can be translated to corresponding OWL models for reasoning and analysis
• **OWL is well-suited to building long-term, tool-neutral archives of project and mission designs**
Example of SysML Profile Application
Example of SysML Profile Application
• Try to keep in mind the idea of classifying things and their relationships with types that are meaningful for space flight in general and JEO in particular
• These classifications are a natural extension of the basic vocabulary of SysML
• They enable the reasoning that is essential for an undertaking of the complexity of space flight