Object Management Group (OMG) and Model-based Systems Engineering (MBSE)

Dick Welling
Boeing
28 September 2015
Welling Bio

B.S., Math and Physics, Michigan State, 1970
Army Reserves: Nuclear Wpns Officer, 1986-91
BDM, Emerson Electric: Loran, GPS, avionics 1980-84
McDonnell Douglas/Boeing: 1984-present
  - Life cycle cost analyst: Adv Tactical Fighter, LHX helicopter, AH-64D (Apache)
  - Systems engineer: AH-64D/E
INCOSE: Reviewer for International Symposium MBSE track papers since 2011
OMG: Advanced SysML certification; voting member of the SysML 1.5 Revision Task Force
Today’s discussion

• OMG
• UML
• fUML
• bUML
• Other things

• Goal: Explain why UML (SysML) is important, why it’s the way it is, its weaknesses, and future direction
What OMG says about itself

Global Product Data Interoperability Summit | 2015

• Founded in 1989, the Object Management Group® (OMG®) is an international, open membership, not-for-profit technology standards consortium.

• Membership includes information technology vendors, end users, government agencies, and academia.

• OMG member organizations write, adopt, and maintain its specifications following a mature, open process in Revision Task Forces (RTFs).

• Largely, its about modeling and modeling languages.
Languages, natural and formal

- Language is a symbolic means for communication.
- A language provides grammatical rules for constructing *statements* that communicate meaning.
- *Natural language* rules can change over time.
- *Formal language* rules are constructed artificially to create statements more precise than natural language.
- UML is a formal language, but only to a point.
• “The” UML was originally intended as a graphical modeling language for software.
• Traditionally used for documenting software architectures and developing test cases.
• Language elements and their associations are preserved in a database.
• Theoretical foundation is set theory.
• A supplemental declarative language for constraints: OCL, Object Constraint Language.
• Has become the lingua franca for defining modeling languages of any kind.
• SysML has extended UML itself to general systems.
Need for executable UML

- Graphical models need validation, otherwise they are only pictures (albeit with an underlying database).
- Validity checks limited to *syntax* (well-formedness).
- Goal: Make models executable to check behavior, etc.
- So far, tool vendors have addressed this with proprietary work-arounds.
- Problem: Ultimately, UML is defined by itself; in other words, it has no formal definition.
- Reference implementation is not possible; spec conformance is by inspection.
- Solution: Foundational Subset for Executable UML Models (fUML).
Executable UML: Issues

- Making models detailed enough for machine execution defeats the purpose of models for human communication.
- UML is not specified precisely enough to be executed.
- Graphical modeling notations are not good for detailed programming.
Executable UML: Issue Resolutions

- Making models detailed enough for machine execution defeats the purpose of models for human communication.
  - Executable models can still be more understandable than executable code.
  - Non-executable models are still useful, too.
- UML is not specified precisely enough to be executed.
  - The Foundational UML (fUML) standard specifies precise semantics for an executable subset of UML.
  - fUML version 1.1 formal specification now available.
- Graphical modeling notations are not good for detailed programming.
  - The Action Language for fUML (Alf) standard specifies a textual action language with fUML semantics.
  - Alf Version 1.0 specification now available.
Syntax & Semantics

• A formal language attaches meaning only to statements that are *well formed* (correctly constructed).

• The *syntax* of the language provides the rules for
  • How to construct well-formed statements
  • Equivalently, for validating that a proposed statement is actually well-formed (grammatically correct).

• The *semantics* of the language then provides the specification of the meaning of these well-formed statements *with respect to a certain domain*. 

Semantics of a Foundational Subset for Executable UML Models (FUML), v1.1
Syntax vs. Semantics

- Specifying syntax is relatively straightforward.
- Semantics is rather more problematical.
  - Meaning requires a semantic domain (universe of discourse)
  - Meaning requires an interpretation of syntax
- Interpretation is the mapping of syntax to a semantic domain to determine the truth value of a statement.
- Statement: “Jack owns that house.”
- “Jack” and “house” are interpreted as real world objects; “owns” is interpreted as a legal relationship between them.
- If Jack does, indeed, own that house then we can say the statement is true under that interpretation.

Semantics of a Foundational Subset for Executable UML Models (FUML), v1.1
Example

- A possible semantic domain is the set (class) of all people and houses, plus an ownership relation.

- This model requires that the name of an instance of Person have a String value and it allows the instance to have zero or more houses associated with it.

- This model instance is consistent under that interpretation.

- Or, if the semantic domain is a Java program, then each model class is interpreted as a corresponding Java class.
## The OMG Metamodel Stack

**Global Product Data Interoperability Summit | 2015**

<table>
<thead>
<tr>
<th>Level (M0-M3)</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0</td>
<td>The domain under study (the “objects” of the model)</td>
<td>777, Ford, etc.: 1&lt;sup&gt;st&lt;/sup&gt; order logic</td>
</tr>
<tr>
<td>M1</td>
<td>The user specification (the model)</td>
<td>Planes, cars, etc.; 2&lt;sup&gt;nd&lt;/sup&gt; order logic</td>
</tr>
<tr>
<td>M2</td>
<td>The modeling language specification (the metamodel)</td>
<td>UML, SysML; set theory</td>
</tr>
<tr>
<td>M3</td>
<td>The reflexive meta-modeling language specification (the meta-metamodel)</td>
<td>MOF – Meta-Object Facility; UML subset</td>
</tr>
</tbody>
</table>

**Semantics of a Foundational Subset for Executable UML Models (FUML), v1.1**
UML Infrastructure = Language for UML

- UML (M2) is written in a subset of UML itself (M3), the UML “Infrastructure.”
- To formally define the semantics of MOF (M3), this circularity must be “broken.”
- A fUML goal is to provide a true abstract base formal semantics for the foundation of UML.
fUML Subset Translation Scheme

Currently action functionality

fUML subset criteria
• Compactness
• Ease of translation to common platform languages
• Primitive functionality

E.g., Java

Semantics of a Foundational Subset for Executable UML Models (FUML), v1.1
Some areas of UML (e.g., use case and requirements models) may not be best formalized based on an executable semantics foundation.

- **Composite Structure Semantics**
- **Complete Activity Model Semantics**
- **State Machine Semantics**
- **Interaction Model Semantics**
- **Non-Executable Model Semantics**

**Foundational Semantics**

- **Base Semantics**
  - The *base semantics* of the subset of fUML used in the execution model are specified using formal logic.
  - fUML operational semantics are specified as an *execution model* written in fUML itself.

CLIF & PSL
• The base semantics (bUML) is expressed in axioms of first order logic
  • Common Logic Interchange Format (CLIF), the language in which the axioms are written
  • Process Specification Language (PSL), a foundational axiomatization of processes
• Completely explicit, rather than using text to explain behavior (as in UML).
• Enables automatic determination of whether an execution conforms to the execution model.
• Disadvantage of requiring axioms for the semantic interpretation of all syntactic patterns used in the execution model.

Semantics of a Foundational Subset for Executable UML Models (FUML), v1.1
Long-Term Vision: A Suite of Specifications

- **Executable UML Semantics**
  - Foundational Semantics for UML Model Execution
  - Precise Semantics of Composite Structures
  - Precise Semantics of Activities
  - Precise Semantics of State Machines
  - Precise Semantics of Interactions
  - Precise Semantics of Time
  - ...

- **Action Language for Executable UML (ALF)**
  - Action Language for Structural Modeling
  - Action Language for Composite Structures
  - Action Language for Activities
  - ...

Ed Seidewitz, OMG TC, SE DSIG, Reston VA, 25 March 2015
Long-Term Vision: Domain-Specific Executable UML

- Precise Semantics of Profiles
  - Specification of the semantics, in general, for defining profiles of executable UML
  - Should include a standard framework for defining domain-specific semantics of profiles, as well as abstract and concrete syntax

- Precise Semantics of Specific Profiles
  - Precise Semantics of SysML
  - Precise Semantics of MARTE
  - Precise Semantics of SoaML
  - …

PSCS Specification already provides non-normative, partial examples of extensions for SysML and MARTE semantics.

Ed Seidewitz, OMG TC, SE DSIG, Reston VA, 25 March 2015
Some Implications of fUML

- Complete formal specification of UML (or at least a significant subset)
- Enables a reference implementation
  - Spec compliance can be tested explicitly
  - Test suite should be part of spec
  - “In theory there’s no difference between theory and practice. In practice there is.” – Professor Lawrence P. Berra, NYYU
- Language should become a true “standard”
  - Dialect proliferation reduced
  - Modeling more efficient
- The “mystery” of UML/SysML may be solved
New in SysML 1.4

- Directed Relationship Property Path
- Element Path Multiplicity
- Adjunct Property
- Bound Reference
- End Path Multiplicity
- Nested Connector End
SE DSIG activities

- Property-based requirements
  - Requirement formalization
  - Explicit part of the system model
  - Beyond “trace” relationships
- “SysML 2.0”
  - Address slow adoption of SysML
  - Views SysML as part of an MBSE environment
  - Better define the environment
  - Language features constrained by that environment
  - Redefine the language: no longer a profile of UML

SE DSIG = Systems Engineering Domain Special Interest Group
Related Activities: Open-MBEE and the K-language (JPL)

- MBSE users are starting to perform analysis
  - Most diagrams are class diagrams, which show structure
  - Some diagrams are state machines or other forms of behavior
  - Looking to encode a lot of requirements or constraints

- Formal methods community is all about analysis
  - Static and dynamic analysis
  - Semantic and syntactic analysis
  - Automated theorem proving

C. Delp, et al, K language, JPL,

Our job is to make these worlds become one!
Questions?
The modeling path

• Create models
• Analyze for correctness
• Discover problems at each level
• Eventually produce implementations
• Rinse and repeat…

C. Delp, et al, K language, JPL,
General Purpose vs Domain Specific

- Modelica is based on an object-oriented, declarative language for modeling the physical systems domain.

```
model Capacitor
  parameter Capacitance C;
  Voltage u "Voltage drop between pin_p and pin_n";
  Pin pin_p, pin_n;
  equation
  o = pin_p.o + pin_n.o;
  u = pin_p.v - pin_n.v;
  C * der(u) = pin_p.o;
end capacitor;
```

- AADL also has a well-defined language for real-time embedded systems.

```
process control
Features
  input_speed: in data port speed_data;
  toggle_mode: in event port;
  throttle_cmd: out data port throttle_data;
  error_set: feature group all_errors;
flows
  speed_signal_path: flow path input_speed -> throttle_cmd;
properties
  Period -> 20 ms;
end control;
```

- UML/SysML is general purpose; the others are domain-specific