UML in the .com Enterprise

The Architecture of UML

Steve Cook
Distinguished Engineer
IBM Global Services
Agenda

UML: a family of languages
UML definition
UML variants
UML architecture
In the industry, UML is used for many different purposes

Direct mapping to language implementations (Java, C++, Smalltalk, CORBA IDL and so on)

“round-trip engineering”

“software through pictures”

Directly executable notation (eg xUML)

Language-independent software specification

Architecture description

Process engineering

Website structures

Workflow specification

Business modelling
The different uses of UML can be semantically different and inconsistent

**Extension by new elements**
- tags, stereotypes

**Class diagrams**
- meaning of aggregation, special symbols

**Statecharts**
- what to attach statecharts to, error treatment, relationship of guards to preconditions, inheritance

**Inheritance**
- allowed forms, repeated inheritance, redefinition rules

**Framing rules**
- what does it mean not to specify something?

**Model of time**
- causes of transitions, timeouts, deadlocks, concurrency

**Connecting OCL to diagrams**
- valid places to write expressions, context of expressions

**Programming-language specifics**
- C++, Java, Smalltalk, ...

**Communication paradigms**
- Buffered, synchronous, asynchronous, reliability, security

**Persistence mechanisms**
- Mappings to databases, logging, undo, error recovery
Agenda

UML: a family of languages

UML definition and use

UML variants

UML architecture
Unified Modeling Language

UML is “a language for visualizing, specifying, constructing and documenting the artifacts of software systems”. It describes:

concepts and their relationships (abstract syntax using meta-model, “semantics” in natural language)
diagrams and notation for concepts
interchange format (XMI)
constraint language (OCL)
repository interface (IDL)
The UML definition is deeply confused about the meaning of “semantics”

The Semantics section is mostly about abstract syntax

The Semantics section contains subsections called Semantics

The Notation section contains paragraphs called Semantics

Each of these sections deals, in natural language, with a variety of things:

- statements about the dynamic behaviour of UML models
- statements about the well-formedness of models
- statements about different modelling approaches
Here’s a traditional (denotational) view of the Semantics of textual languages.

Concrete syntax: (3 + 2) * 7

Abstract syntax: \( \text{times}(\text{plus}(3, 2), 7) \)

Semantics: 35

“Semantic function”

\[
E: \text{ExpTree} \rightarrow \text{Number}
\]

- \( E (\text{plus} (e_1, e_2)) = E (e_1) + E (e_2) \)
- \( E (\text{times} (e_1, e_2)) = E (e_1) \times E (e_2) \)
- \( E (3) = 3 \), etc

“Semantic domain”
How can we apply this approach to UML?

Concrete syntax

Abstract syntax

Semantics

Diagrams

XMI

IDL

Metamodel

instances

map

“Semantic domain”
Agenda

UML: a family of languages

UML definition

UML variants

UML architecture
The OMG concept of “profile” recognises the need for UML variants

A UML Profile (according to the Profiles Green Paper OMG ad/99-12-32) :-

Consistently specialises the UML metamodel
Uses only UML extension mechanisms
    stereotypes, tagged values, constraints
Is interchangeable using existing (XMI) mechanisms
Defines a subset of UML metaclasses
References domain-specific libraries
Can be specialised and composed
Can be associated with UML packages
Formalises the notion of “tagged value”
Example profile: UML for CORBA
UML Profiles are only one of the UML variant mechanisms currently in use

UML Profiles are limited

No additional classes in meta-model, only stereotypes
No associations between stereotypes
Only iconic modification of notation
• thus can be supported easily by tools

A different approach has been used for the OMG Common Warehouse Metadata standard

Define, using MOF directly, full subclasses of UML metamodel classes, and MOF associations between those subclasses
Mapping to notation is formally undefined
Agenda

UML: a family of languages

UML definition

UML variants

UML architecture
The UML 2.0 Infrastructure RFP calls for:

**Architectural alignment and restructuring**
- strict alignment with 4-layer model
- make MOF a subset of UML
- restructure the metamodel in order to separate concerns
- identify “semantic variation points”
- backwards compatible with XMI 1.x

**Extensibility**
- specify profiles
- specify “first class extensions”
IBM has funded a feasibility study by pUML (precise UML group)

See www.puml.org for the document “A Feasibility Study in Rearchitecting UML as a Family of Languages using a Precise OO Meta-Modeling Approach”, (Clark, Evans, Kent, Brodsky, Cook) and associated tools

The study proposes a new meta-modelling facility (MMF) containing:

- Meta-Modelling Language (MML)
- Meta-Modelling Tools (MMT): a satisfaction checker - does instance X satisfy constraint C from model M?
  - check that a model satisfies its metamodel
  - check that a metamodel satisfies the MML rules
  - check that MML satisfies the MML rules
The structure of MML

(from Logical View)

UML

staticCore

associations

reflection
datatypes
model Management
constraints

mml

methods
The internal structure of the staticCore package

```
staticCore
(from UML)

staticCore.model

staticCore.model.concepts
(from staticCore.model)

staticCore.semantics

staticCore.instance

staticCore.instance.concepts
(from staticCore.instance)
```
The staticCore.model.concepts package

- **ModelElement**
  - name : String

- **Container**

- **Generalisable**
  - conformsTo()
  - +attributes
  - +paren tally
  - +generalisations
  - +specialisations

- **Generalisation**
  - +child
  - +parent
  - +generalisations
  - +specialisations
  - +elements

- **Classifier**
  - isAbstract : Boolean

- **Class**

- **Attribute**
  - +attributes
The staticCore.instance.concepts package

```
InstanceElement

+elements

Instance

+value

Object

+slots

Slot
```
The staticCore.semantics package
OCL constraints are used to specify semantics

context UML.staticCore.semantics.Instance inv:
satisfies(c: Classifier): Boolean
if self.of = c then
    of.allContents() -> forall(e1 |
        elements -> exists(e2 |
            e1.name = e2.name and
            e2.satisfies(e1)))
else false
endif
The structure of MML defines a systematic method for extending MML

1. determine whether the model element is a subclass of Classifier, i.e. exhibits the properties of a generalisable container;
2. if so, subclass the model element from Classifier in the model.concepts package;
3. constrain the model element’s contents to be those of the attribute ‘elements’;
4. in the instance.concepts package, identify or add a new instance subclass which is an instance of the classifier;
5. in the semantics package, link them by subclassing the ‘of/instances’ association;
6. for each element of the new model element repeat steps 4-5;
7. determine any dependencies between instances and their elements, and specify these using appropriate constraints.
MML includes a complete model of OCL - concepts, instances and semantics.
UML is positioned in the OMG’s “4-layer architecture”

Metametamodel

<table>
<thead>
<tr>
<th>M3</th>
<th>Meta-Object Facility</th>
</tr>
</thead>
</table>

Metamodel

<table>
<thead>
<tr>
<th>M2</th>
<th>UML, CWM, SPE</th>
</tr>
</thead>
</table>

Model

<table>
<thead>
<tr>
<th>M1</th>
<th>My model</th>
</tr>
</thead>
</table>

User objects

<table>
<thead>
<tr>
<th>M0</th>
<th>What I’m modelling</th>
</tr>
</thead>
</table>
But the current architecture of UML / MOF is not well-organised
We’re working on a possible new architecture for UML / MOF

- UML 2 (concepts, syntax, semantics)
- MML
- OCL
- MOF’
- XMI, IDL
- CWM 2
- Profile(X)

Relationships:
- UML 2 extends (M2) MOF’
- MOF’ defines XMI, IDL
- OCL instantiates MOF’
- Profile(X) extends (M1) CWM 2
Would this answer the RFP?

Architectural alignment and restructuring

*strict alignment with 4-layer model*
- Yes, if MML redefines M3

*make MOF a subset of UML*
- Yes, with some changes

*restructure the metamodel in order to separate concerns*
- Yes, especially by improving / clarifying “package extension” semantics

*identify “semantic variation points”*
- Every metaclass is a SVP subject to constraints

*backwards compatible with XMI*
- MML formalises syntax, it can define mappings to any version of XMI

Extensibility

*specify profiles*
- Profiles apply simple MML extensions encoded at M1

*specify “first class extensions”*
- MML is specifically designed to do this, including semantics
Conclusions

UML is a family of languages, not a single language

The architecture of the UML family needs improvement:

  a clear approach to semantics

  a clear approach to UML variants

  simplification and separation of concerns

UML 2.0 is a major opportunity to introduce such improvements