A Systematic Approach to Domain-Specific Language Design Using UML

By Bran Selic

Presenter

Henry Zaccak, MHI Candidate, HBSc

March 19 2012

Outline

- The Author
- Background Information
- The Problem
- The Solution
- Domain Specific Model Languages
    - DSML Challenges

Outline II

- Quick History of UML
- UML 2
  - Stereotypes
  - Profiles
- Using UML Profiles to Define a DSML
- Conclusion
- Discussion

About Bran Selic (Current)

- Adjunct Professor – U. of Toronto (3Y+)
- Adjunct Professor – Carleton U. (12Y)
- Adjunct Research Scientist–Simula Research Lab (2Y)
- Director of Advanced Technology – Zeligsoft (3Y)
- President / Founder – Malina Software Corp (4Y+)

Source http://www.linkedin.com/pub/bran-selic/0/5b/a4

About Bran Selic (Past)

- Distinguished Engineer – IBM Canada (Many many years, retired in 2007)

Background

- Increasing knowledge and experience brings greater domain specialization
- In software engineering, this can be seen in the development of many programming languages (Fortran, Cobol, ..etc) built for specific domains (numerical computing, data processing, statistical processing..etc.)

Background II

- An essential feature of all these specialized domain languages concepts is they are rendered as first order language constructs as opposed to rendered through a combination of one or more general constructs
- This greatly eases a programmer’s task as it enables direct expression of problem-domain patterns and ideas

The Problem

- Although new DSL's are still being developed, the vast majority of current software is written in standard general purpose languages (Java, C++, C#)
- This is due to:
    - Lack of support for highly specific languages (compilers, editors, debuggers, linkers..etc)

The Problem II

- Supported tools are not well designed, user friendly
- Little economic incentive for major tool vendors to invest in specialized languages for a small community of users
- General purpose programming languages has a readily available trained user base

The Problem III

  - DSL’s lack pre-packed libraries
  - Open source software community uses generic programming languages
- Instead of replying on DSL’s many organizations rely on Domain specific program libraries (written in general purpose languages).
  - Provide a “similar” feel to first order language
  - ..but tied to the same programming language

The Problem (Summary)

We want to define an expressive domain specific language… but we want to keep the benefits of existing libraries / frameworks

The Solution

Use the UML Profile mechanism to define your “expressive” domain specific modeling language!

…but your DSL must be conformant to UML (not in conflict with UML semantics / syntax).

Domain Specific Modeling Languages

- Modeling language design is still an emerging discipline with few proven guidelines / patterns
- Some theory can be shared with programming language design but some challenges are unique

DSML Challenges

- The need to support different levels of precision: different degree of formality
- The need to represent multiple consistent views of certain elements of the model
- The graph-like nature of most modeling languages can’t be easily reduced to linear text-based representation

DSML Approaches

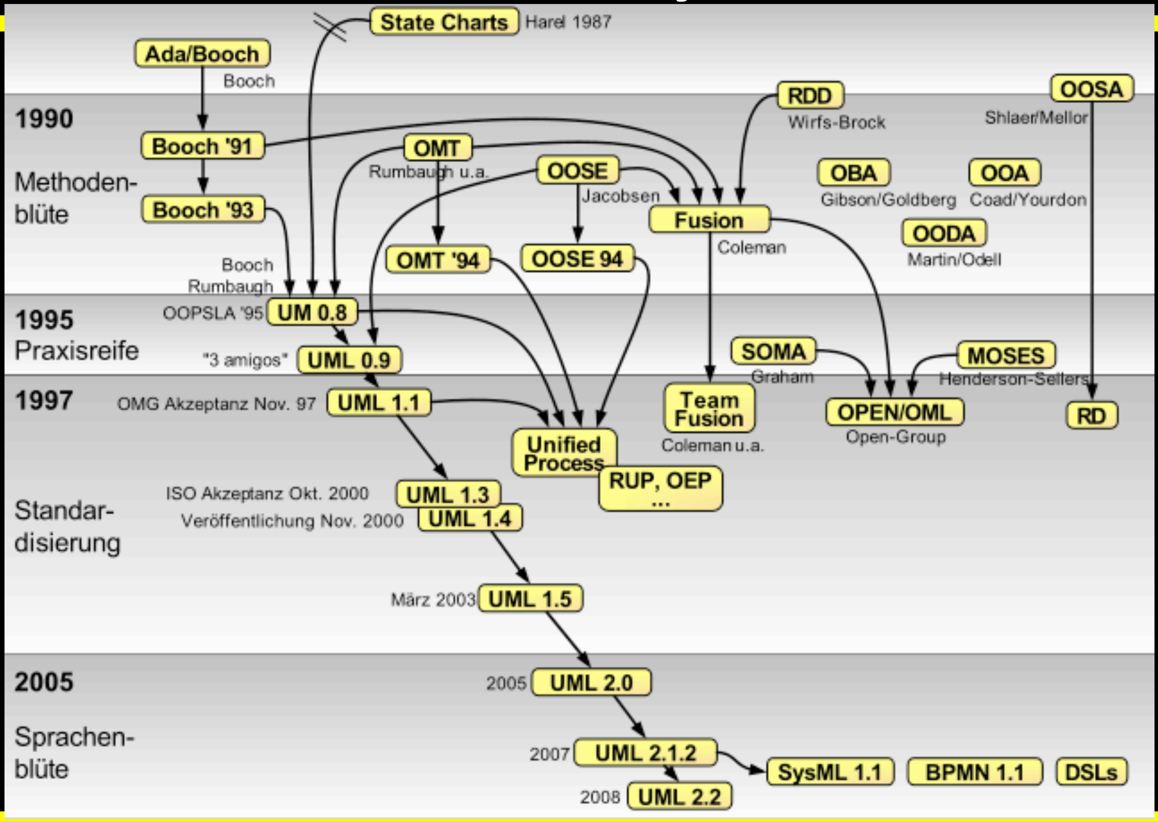
1. Refinement of an existing more general modeling language by specializing some of its general constructs to represent your domain specific concepts
2. Extension of an existing modeling language by supplementing it with fresh domain specific constructs

DSML Approaches II

3. Definition of a new modeling language from scratch!

- The first option seems to be the most practical and most cost-effective approach. (Tool reuse!)

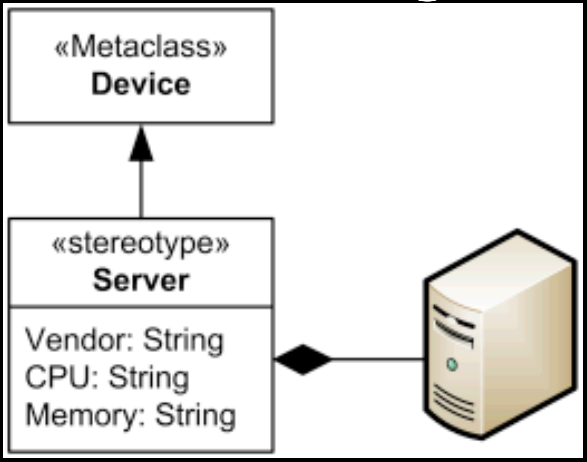
Quick History of UML



Source http://en.wikipedia.org/wiki/File:OO-historie-2.svg

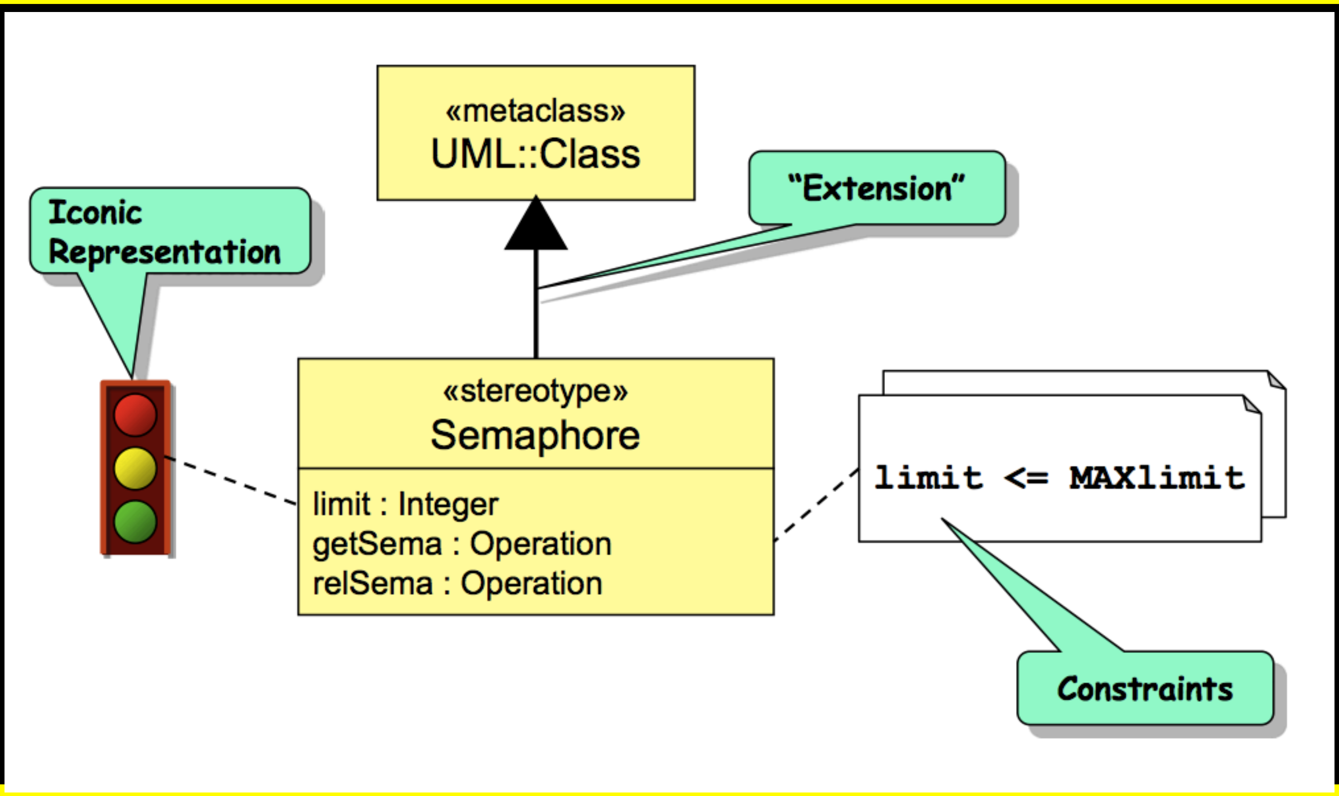
UML 2 - Stereotypes

- Stereotypes allow designers to extend the vocabulary of UML in order to create new model elements from existing ones but specialized



http://www.uml-diagrams.org/profile-diagrams.html#stereotype

UML 2 - Stereotypes II



Source: The Theory || and Practice of Modeling Language Design for Model-Based Software Engineering - Bran Selic

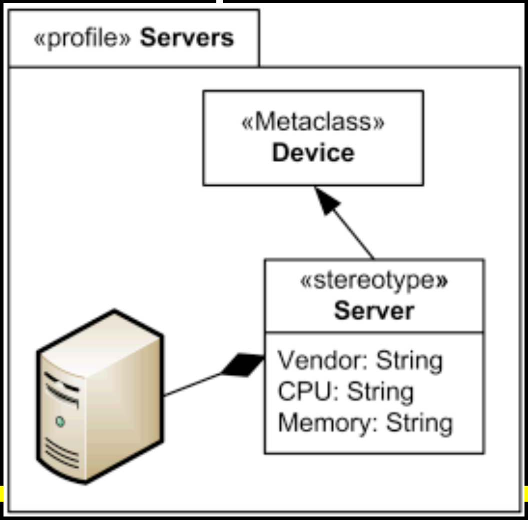
UML 2 - Stereotypes III

Why not define a new element of a UML model in the standard way?

1. To allow tool implementers flexibility in choosing their preferred implementation
2. The need to support viewpoints, which require the ability to dynamically apply/ un-apply stereotypes

UML 2 - Profiles

Profiles are packages of stereotypes, model libraries and metamodels which defines domain specific concepts and relationships



UML 2 – Profiles II

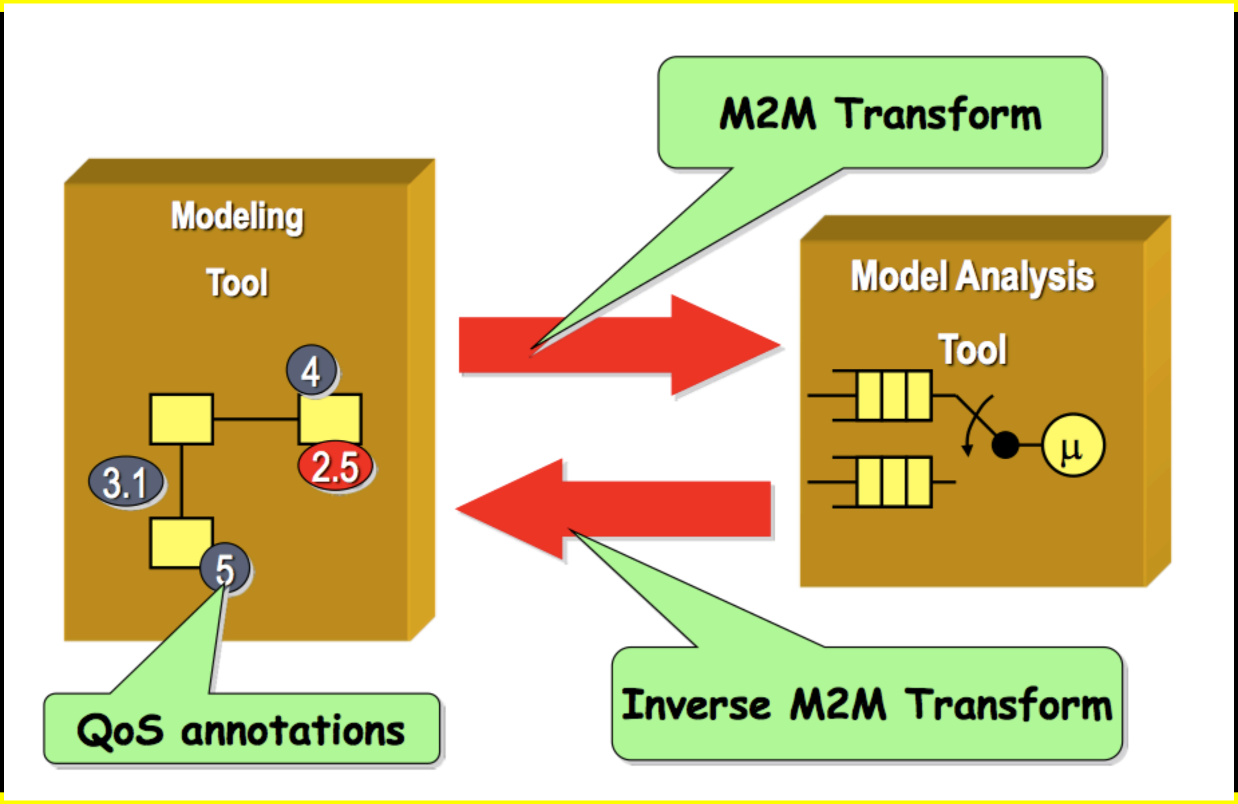
Two ways to use a profile:

- Define a DSML (create all your domain specific concepts)
- Define a domain specific viewpoint (ie. Using a performance viewpoint to measure the performance of your model)

UML 2 – Profiles III

- Can have formal constraints written in OCL
- Can apply /un-apply stereotypes profiles to UML models (can see model from different viewpoints)
- Can create relationships that don’t exist in the metamodel

UML 2 – Performance Profile Example



Source: The Theory || and Practice of Modeling Language Design for Model-Based Software Engineering - Bran Selic

Using UML Profiles to define a DSML

Approach:

1. Create a domain model (or metamodel)
2. Map the domain model to a Profile (Map to UML)

Using UML Profiles to define a DSML II

Create a domain model:

1. A set of fundamental language constructs
2. A set of valid relationships
3. A set of constraints
4. Concrete syntax or notation
5. The semantics of meaning of the language

Using UML Profiles to define a DSML III

Map the domain model to a profile :

1. Select a base UML class whose semantics are closest to the semantics of the domain concept
2. Check all the constraints that apply to the selected base metaclass to ensure no conflicting constraints

Using UML Profiles to define a DSML III

Map the domain model to a profile :

3. Check to determine if any of the attributes of the selected base class need to be refined.
4. Check to determine if the selected base class has no conflicting associations to other classes.

Conclusion

- The UML 2 Profile mechanism can provide powerful capability to define DSMLs
- Allows reuse of standard UML tools
- Author provided a “methodology” to produce technically correct profiles
- … but still need a proper theory for model language design

Discussion

- What type of DSMLs can we model with profiles? What type wouldn’t be so useful?
- Can our model use viewpoints for different code generation paths?
- Can all programming languages be remodeled in profiles?
- What is the future of modeling?