

# Domain Modeling and Transformation Specification

Eric Engstrom  
Jorn Bettin  
Ghica Van Emde Boas  
Dean Wampler  
Wei Zhao  
Krzysztof Czarnecki

## Topics

1. Transformation languages
  - a. Visual/ Textual
  - b. Different types Tree re
  - c. Power
  - d. Usability
  - e. Scalability
    - i. How does visual scale
  - f. Composition
  - g. Modularization
  - h. Transformation complexity
  - i. Template language? (What role)
    - i. Linear / graph
    - ii. Typed / untyped language (c++ templates vs Template Language TL)
  - j. Recording to transformation
2. Domain modeling
  - a. Breadth of domain
  - b. Family modeling
  - c. Separation of concerns
  - d. Life cycle of the MDA (business to design to develop)
  - e. Aspects
3. Platform modeling
  - a. What is a PSM vs PIM.
  - b. Executable
  - c. Different layers of models w/wo different aspects
4. Domain Language and Transformation coupling
5. Output Executable
  - a. Quality
  - b. Meet requirements
6. Big picture of MDA
  - a. Guidelines (given by MDA)
  - b. Development process (given by MDA)
  - c. Tools (needs to be developed)
  - d. Techniques (needs to be developed)
7. Component Acquisition business

## Why LEGO Blocks don't work

### Topics

1. Software components are not physical things
2. Parameterization
3. Components as parameters
4. Components don't work because of lack of specification?
5. Is Adaptability domain specific?
6. Does static typing impede adaptability?
  - Adaptability is moving components from one context to another
  - A component reacts to context change and adapts itself (internal)
7. Component should be black box?
8. Layering
9. Runtime vs. construction time adaptability
10. Type objects are a runtime construct versus simple classes generated at compile time
11. System functions, ui functions, etc MDA....
12. Annotate models with extra information about mappings
13. Dead specification removal
14. No need for inheritance if code is synthesized...
15. Has-a is more reusable (pluggable)
16. Is inheritance generalization or specialization?
17. Difference in Context is the common culprit
18. We can't envision every possible variation up front
19. Wrapping, extending
20. Dynamic code loading, runtime reflection
21. Aspects!
22. What's a component in the context of MDA
  - well defined interface
  - well specified interaction
  - configuration port?
23. Reflection is not the bottleneck in most systems
24. In MDA, a components is most probably OMG component specification!
25. Black boxes CAN be adaptive! It can have configuration parameters that completely change its behavior. State models inside components.
26. RT needs only and/or tree as hierarchical decomposition

OMG is Oh My God ☺

27. Yoder: There is no generic domain independent adaptivity "language"
28. Yunker: The more fine-grained the components the more intelligence is needed in the assembly process.
29. Adaptability is a recursive problem
30. We don't have tools that support it.
31. At what point does MDA meet up with a component infrastructure?
32. Should MDA generate every little aspect of the entire system?
33. Platform specific services are not generated, nor is it needed.

34. Your generated system has to interact with other non generated systems which are viewed as “interfaces” or services, but generated system and thus the generator has to know about them.
35. It all hinges on the definition of “Component” and “Adaptability”
36. A component has communication interfaces and configuration interfaces.
37. Separation of concerns and put them in a container (security, fault tolerance, load balancing, etc. systemic aspects). Keep them out of the component. This way a component can adapt to those concerns because the container allows it to.
38. Adaptability of systemic aspects is relatively well understood and implementable with existing technology. Adaptability of business functionality is not so straightforward.
39.  $N + M$  versus  $N * M$  problem

## Conclusion

LEGO BLOCKS COULD WELL WORK IF they are adaptive to context. Adaptivity of systems aspects is easier to achieve than that of business aspects.

## Aspects

### Summary

An important goal of modeling is to deal with the non-linearity in software: the size in the change in requirements should be proportional to the change in the implementation. The word “aspect” may not have a precise definition, but it generally refers to design decisions that are non-local relative to a given viewpoint

A model is an instance of a meta-model, which defines its notation and semantics. There is a stack of meta-models which usually stop after 3 or 4 levels, and

Current practice is that models are integrated by a “model weaver” which interprets/compiles the models and implements their semantics. The meta-models are generally implicit.

Creating modular “weavers” which allow composition of models based on their meta-models is an important research direction. Creating correspondence models that define the semantics of model integration is one possible approach.

Models can be used at runtime: dynamic interpretation of models that can be modified by users and reflection to explain programs behavior.

Is there any reason to keep the models around at runtime

- 1) If the user wants to change the models
- 2) Reflection. Context-dependent help that is based on the state of the model, not just on the runtime information. In Word you can only print one document at a time. When you want to explain why the Print menu is grayed out.

### What is an aspect?

System

- Data
- UI
- process
- workflow
- security
- workflow

Viewpoint?

Can you define a system without a dominant decomposition?

Is exception handling componentizable.

Setting up a dominant decomposition

Aspect-oriented programming

Aspects are things that cross-cut

### **How do you combine models?**

The meta-model semantics defines the behavior.  
Single compiler that knows about all the models.

Each model is written the language of the meta-model.

Java program P (meta model is Java)  
UML model of the program (UML is model)  
The trace is a specific execution of the system.

What is static system? The program is static  
Models are declarative, but they define

With a GUI of the system, the drawing is a model?  
will the meta-model talk about position, layout, etc and the widgets.  
to create a meta-model,

UML 2.0  
separation of content and presentation in modeling

### **Questions**

Most software development begins with use cases, and CRC cards.

Are they models? Sure, they have structure, so they are models but they are not integrated models:

- if you define them precisely as specifications (pre/post conditions) then you do have a precise model. When you have precise definitions, you can advance very rapidly.
- If you want to extend/change the meta-model, then you may be ok, but if you want to change the domain, then you have a lot of work.

The purpose of the compiler is to restrict the set of allowed description so that they can control what people say?

What are the hard questions:

How do you provide models, you may lose expressiveness.

If you have an existing system, how do you go about integrating an new aspect. Had to tag on a specific kind of event logging. Went through the system and added before/after events in a system. Also useful for business logic.

Had to add multi-national to the existing system. Change the compiler and then everything.

Different kind of modularity: instead of splitting system up into components, split it into a model compiler and then a set of models.

Microsoft has a tool to generate C# from Java. In Java if you have a method that is visible but the visibility rule is different. Their translation is more restrictive than Java.

You could also go from Java to UML, then UML -> UML and then from UML to C#.

There is no modularity at the meta-model level.

We have some customers that will not allow JVM, others will not allow anything from Microsoft. Is there any way to do joint development. He defined a transformation system to move from Smalltalk V to smalltalk Parcplace.

To be able to define the combination of models using meta-models, you need a correspondence model.

mode: m  
    grammar G defines language L  
    interpretation: f  
m elem L  
f(m) is interpretation of m

then you create the meta-grammar and meta-interpretation

A → MA

B → MB

you need a MAB that links the two together. Can you factor out the meta-models.