Industrialized Software
Open questions or consensus?

The Third KISS Workshop
25 October 2009 @ OOPSLA
Orlando FL, USA
Use cases vs. meta modelling?

- If deep domain expertise is available:
  - ★ Start with the definition of meta models
  - ★ Validate meta models with a sufficient number of sample models
  - ★ Validate the domain terminology by using it to write use cases for modelling language users
  - ★ Validate modelling languages and sample models by using them to generate a working implementation

- If deep domain expertise is lacking:
  - ★ Start with writing use cases to identify candidate domain terminology
  - ★ Validate use cases by manually developing a first implementation
  - ★ Use a second and third implementation to incrementally identify patterns, and use domain analysis to distill first meta models
Do modelling languages differ from coding languages?

**DSM** or **DSL**, what is in a name?

Observation: Given that in the context of software the term coding is often used interchangeably with programming, it is instructive to compare the dictionary definitions of *to code* and *to model* to understand the not-so-subtle difference in intent:

- **to code**: express (a statement or communication) in an indirect or euphemistic way
- **to model**: devise a representation, especially a mathematical one of (a phenomenon or system)

Coding can be viewed as having to deal with someone else’s representation (program notation or otherwise). This is exactly what happens when we work with third party implementation technologies and when mapping to such technologies in a template language.

Modelling can be viewed as dealing with a representation that is fit for purpose. This is what happens when we capture knowledge in a domain specific language that is grounded in established domain terminology.
Notation design elements and options

- Textual syntax
- Form based representation
- Tree based representation
- Boxes (shapes) and lines paradigm
- Nested boxes or element to child-diagram “drill-down”
- Two dimensional arrangements of domain-specific symbols
- Animated diagrams
- Combinations of the above
- More? Applicability guidelines?
Domain engineering process

Domain Analysis & Design

- Validate languages
- Design languages
- Analyse decisions
- Validate reference application

Domain Implementation

- Extract templates or develop execution engine
- Develop platform
- Develop reference implementation
Domain engineering roles

Customer

Domain Engineer
  - Language Designer
  - Domain Analysis Facilitator

Domain Expert

Domain Architecture Developer
  - Product Architect
  - Prototype Developer
  - Technology Specialist
  - Transformation Developer
Difference between a model editor & an ordinary application

- The user of a model editor is always aware that (s)he is manipulating a model.
- The model conforms to a formal notation that the user is familiar with.
- The model has well-defined semantics which are provided via a generator, an interpreter, or other component that relies on the meta model corresponding to the formal notation to navigate the model.
- When using and/or implementing an “ordinary” application:
  - The user relies on a fuzzy mental model of what the application does, he is not dealing with a familiar/intuitive formal notation.
  - The application developer relies on a fuzzy mental model of the concepts that constitute the terminology of the user.
  - Hence the implementation can’t contain any formal mapping between problem space and solution space.
Unlocking the full potential of model driven software

- Using **model driven techniques to implement applications** conceived via a traditional requirements analysis process **only requires a paradigm shift in software engineering**

  ★ software development productivity typically increases by a factor of 2 or more

  ★ Quality rises as whole classes of defects can be eliminated systematically

- Using **domain analysis to uncover deep domain knowledge** and conceive model driven applications **requires a paradigm shift in business modeling and requirements engineering**

  ★ Potential productivity and quality improvements by an order of magnitude
Thank you!

Jorn Bettin
jbe @ sofismo.ch
Skype jorn_bettin
+41 62 891 0987