Advanced SW Engineering: Model-driven Development

Uwe Zdun
Software Architecture Research Group
Faculty of Computer Science
University of Vienna
http://cs.univie.ac.at/swa

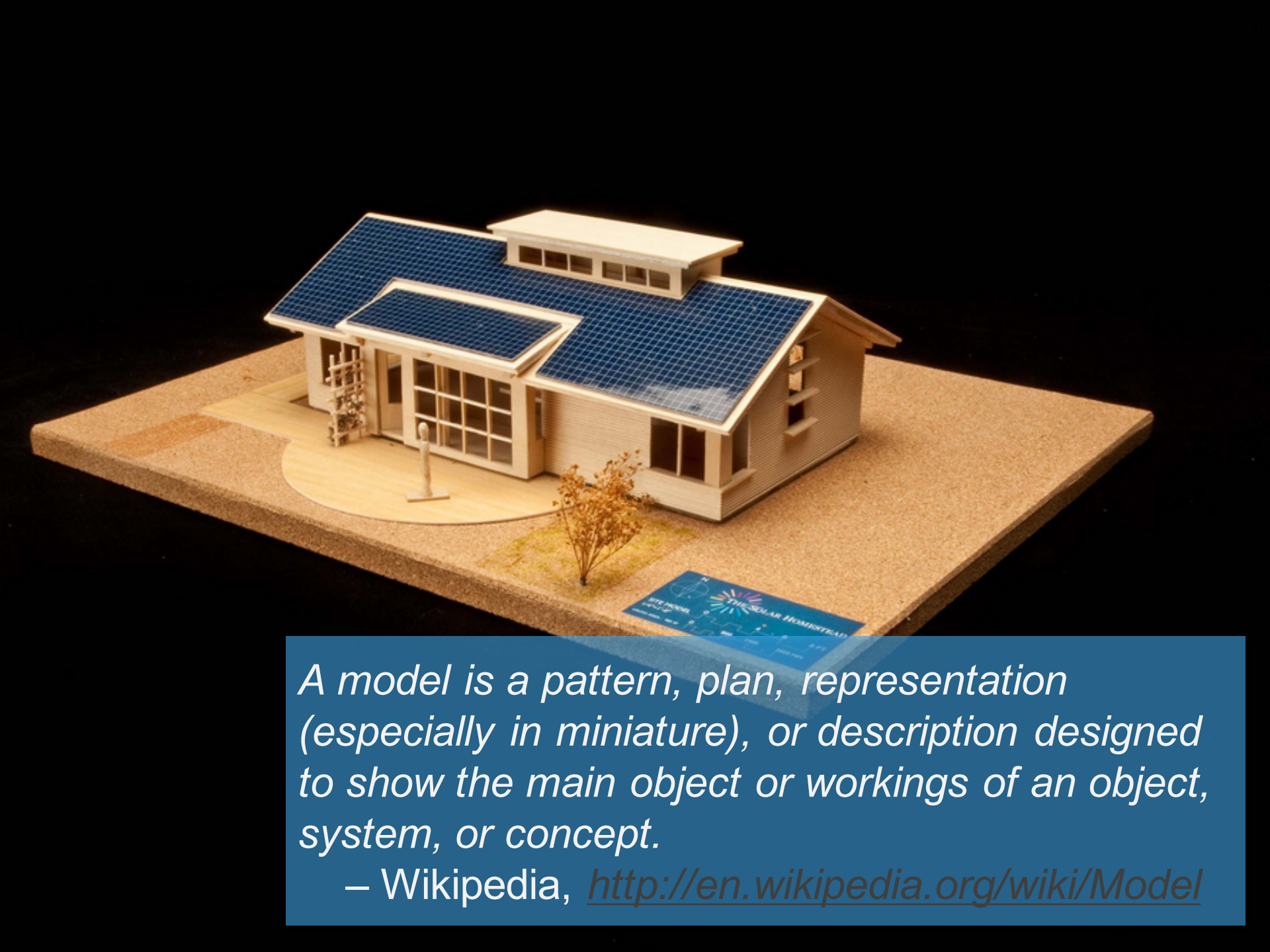
MDD

What is model-driven development?

- Two important terms:
  - Model
  - (Software) Development
- Synonyms / Related Terms:
  - Model-driven Software Development (MDSD)
  - Model-driven Architecture (MDA)
    - MDA of the OMG is one specific flavor of model-driven software development (MDSD or MDD)
  - Model-driven Engineering (MDE)

Main Idea

Models are not only used for documentation and communication, but as central artifacts of the software development process



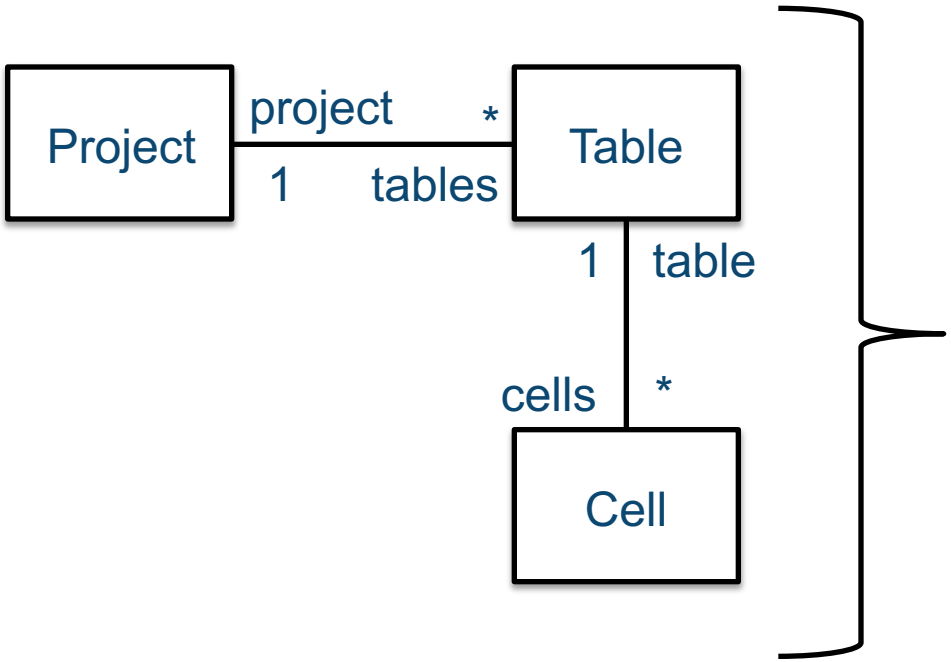
A model is a pattern, plan, representation (especially in miniature), or description designed to show the main object or workings of an object, system, or concept.

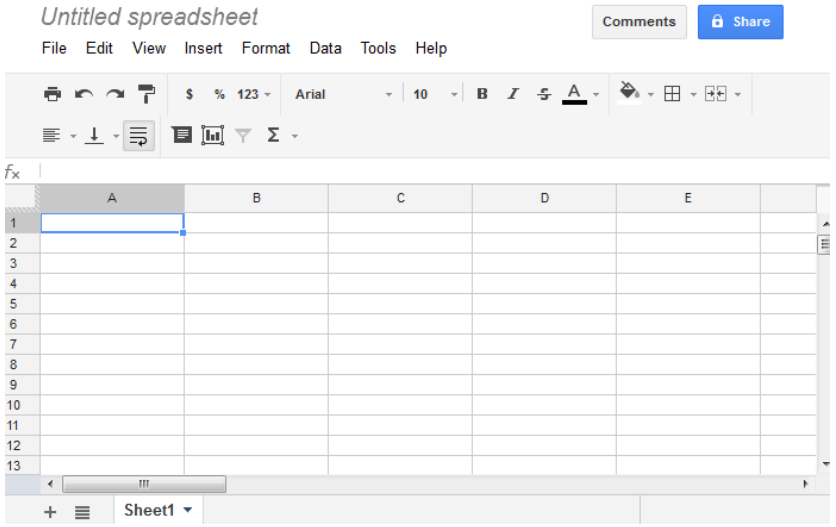
– Wikipedia, *http://en.wikipedia.org/wiki/Model*

Models in software development

In software development a model is an abstract representation of a system.

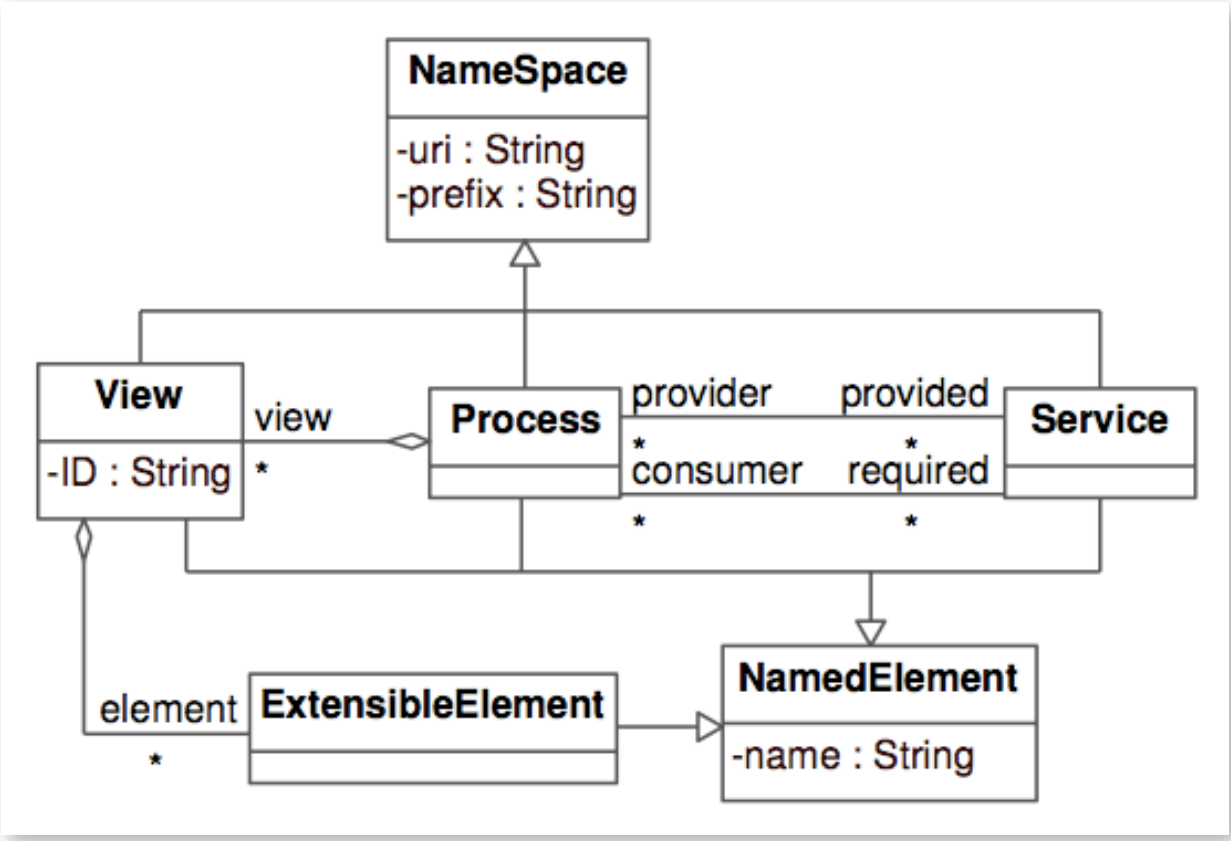
– Vogel et al.: Software-Architektur



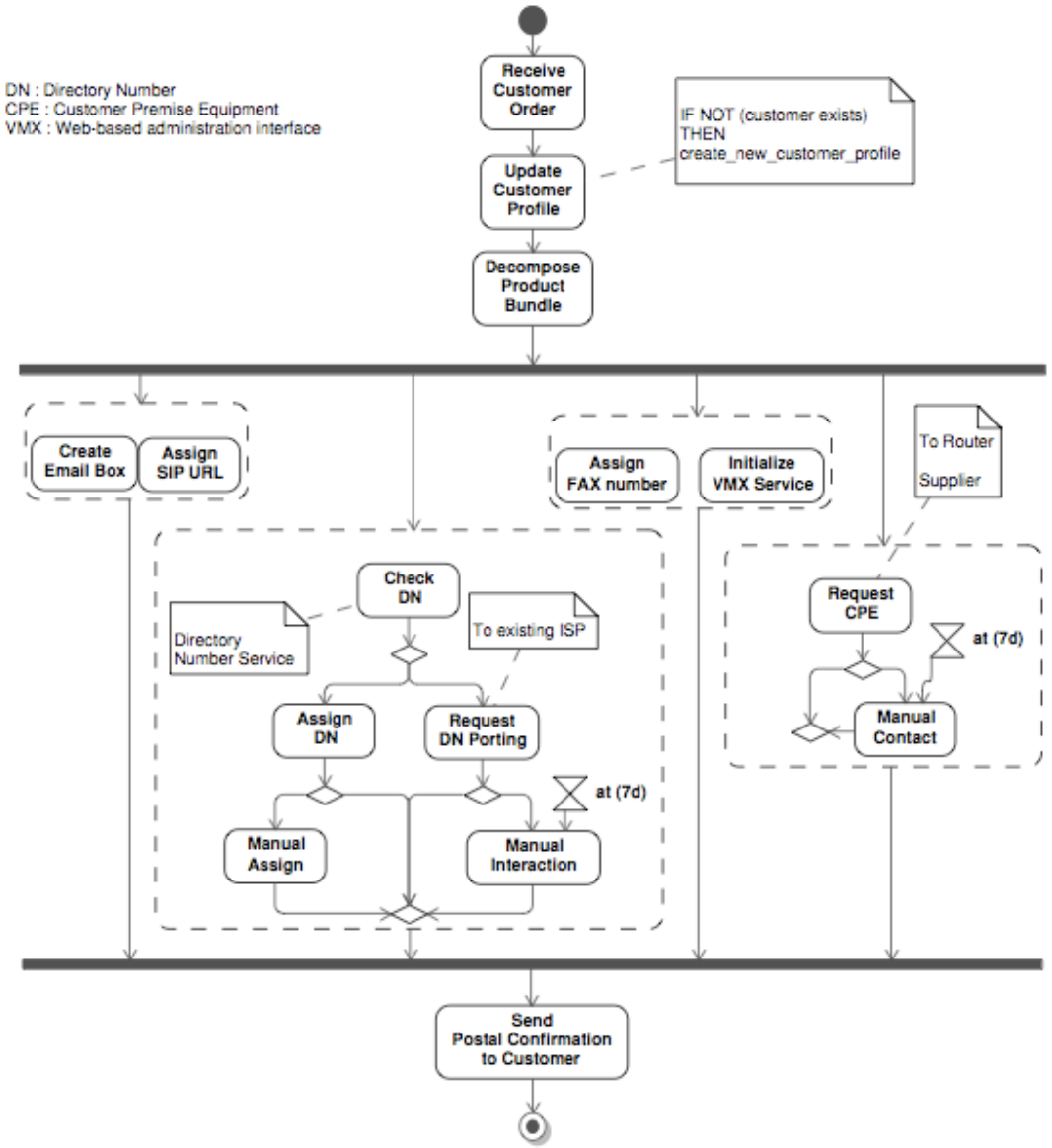


MANY KINDS OF MODELS ARE USED IN SOFTWARE DEVELOPMENT

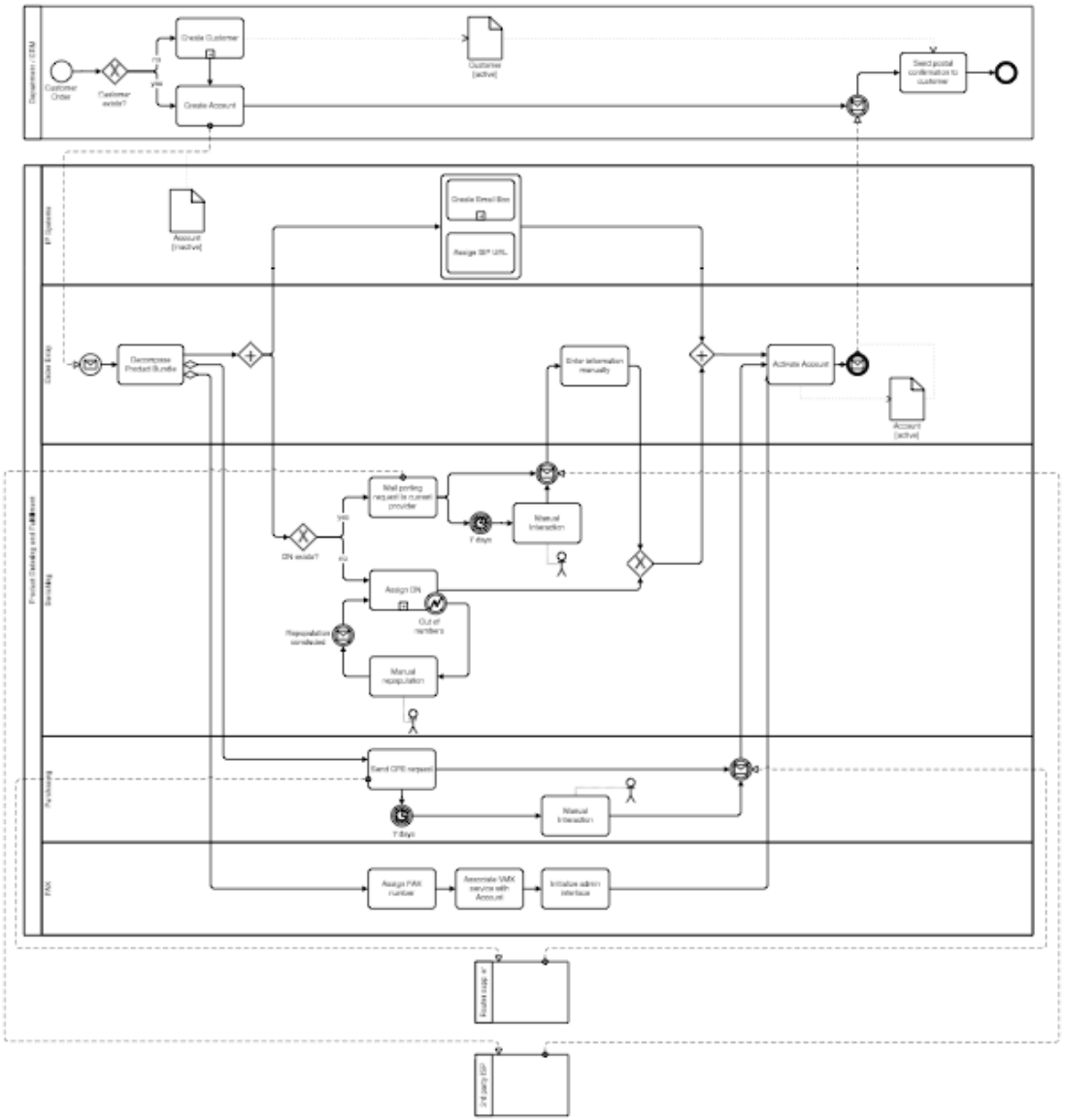
Example: A Meta-model



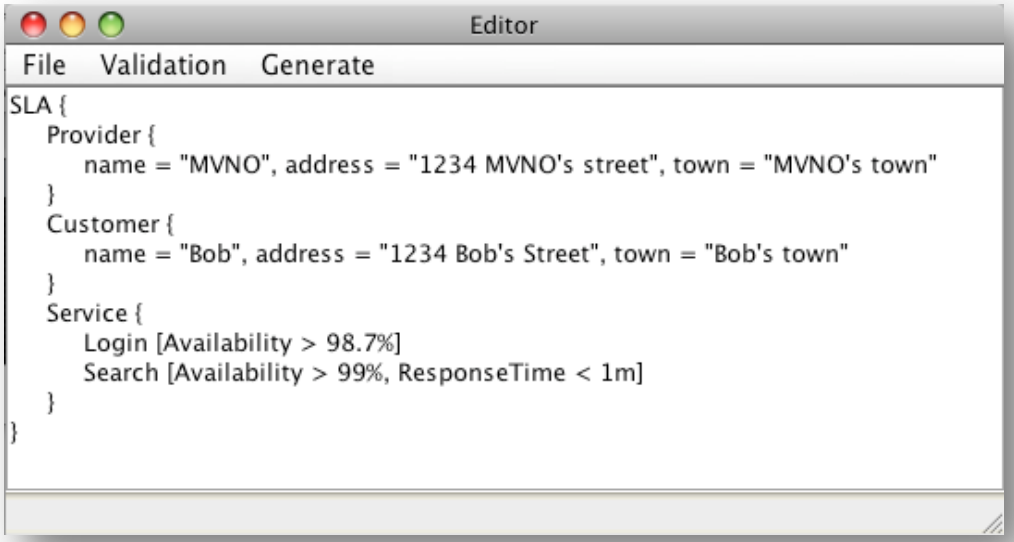
Example: A Process Model



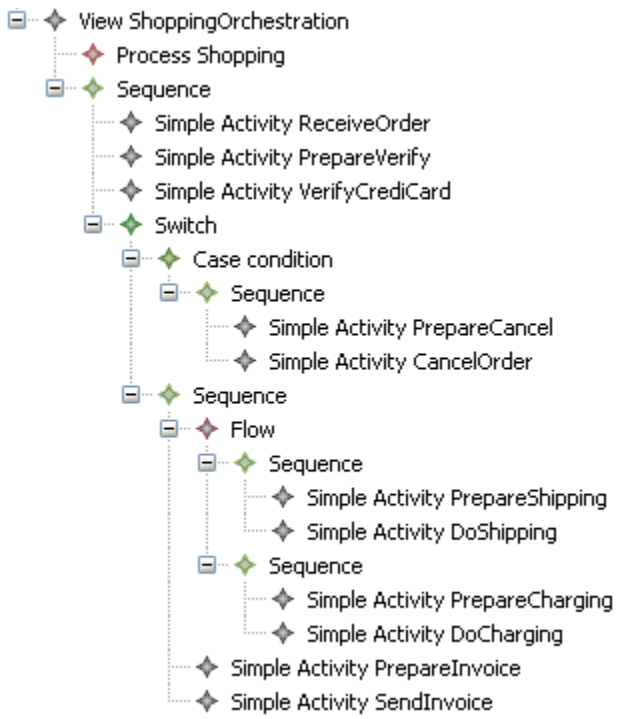
Example: Another Process Model

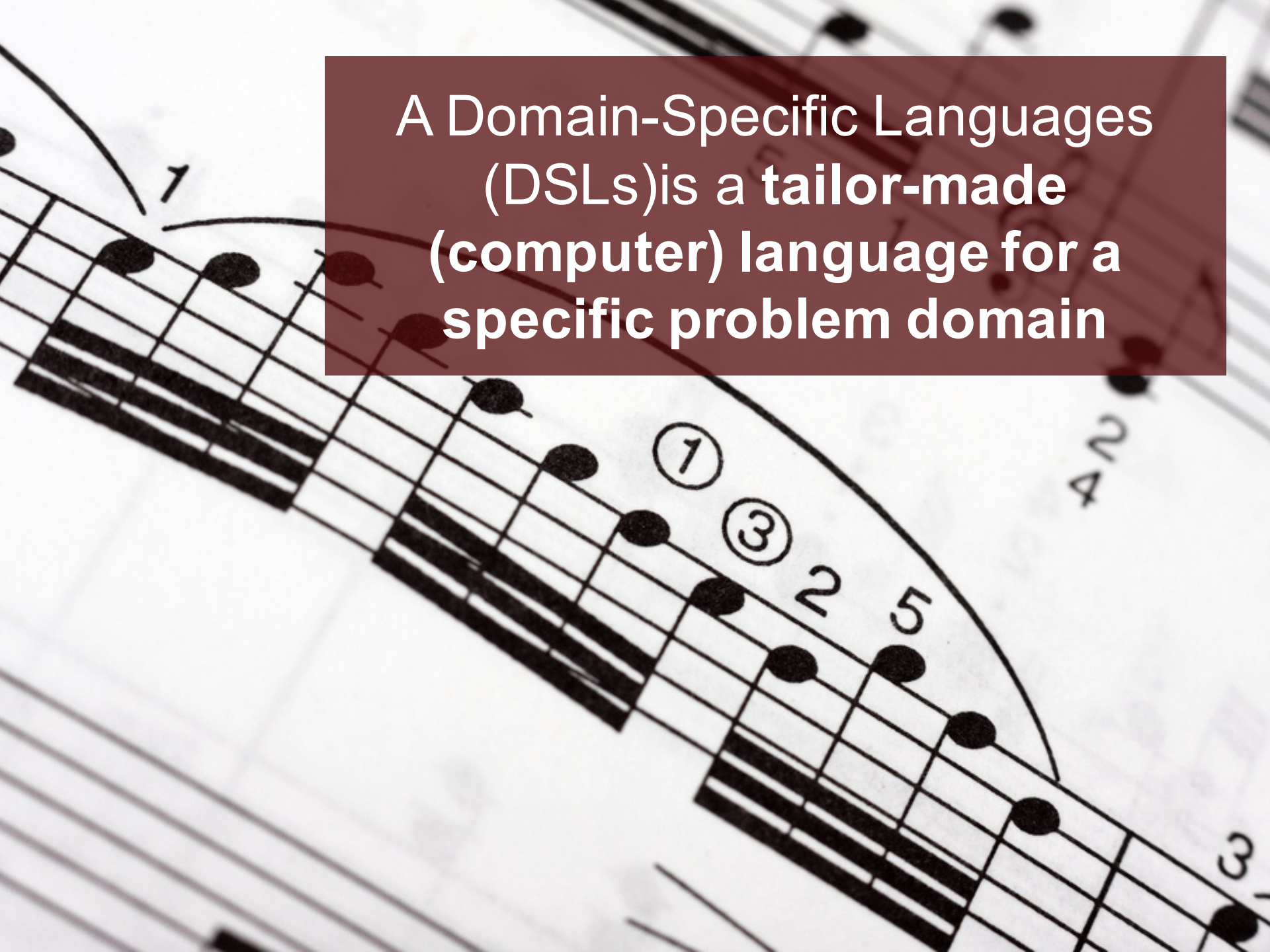


Example: A Textual Model



Example: A Model in a Tree-Based Editor




A Domain-Specific Languages (DSLs)is a **tailor-made (computer) language for a specific problem domain**

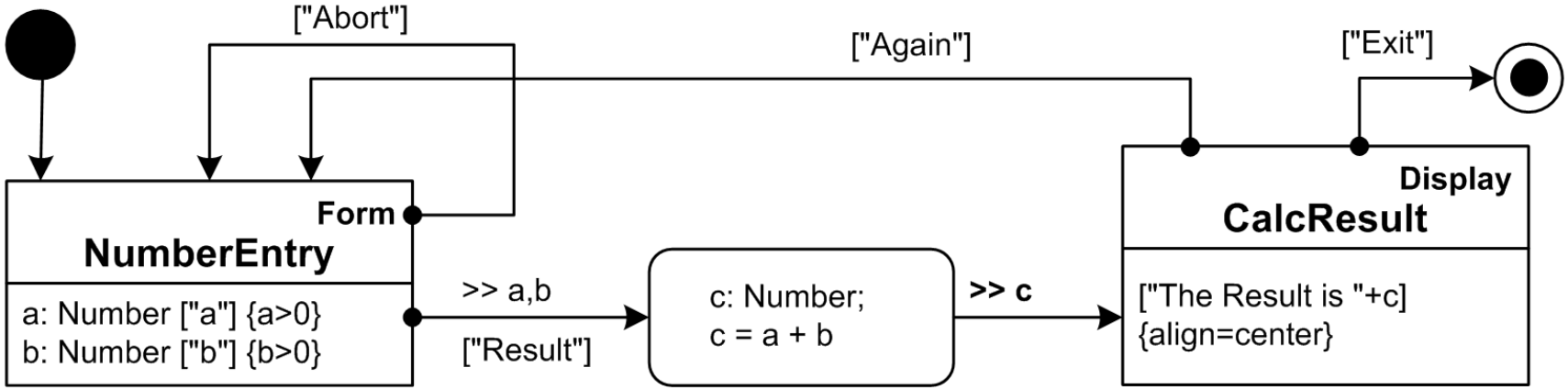
Domain-Specific Languages (DSLs)

- DSL are **special-purpose languages**
  - DSLs differ from general purpose languages (GPL), such as C, C#, Java, Perl, Ruby, or Tcl, that can be applied to arbitrary problem domains
  - DSLs are often not Turing complete and only provide abstractions suitable for one particular problem domain
- This specialization results **in significant gains in expressiveness** and **ease of use**
  - Often **less code** is needed to achieve a result
  - Often DSL code is rather **easy to read and comprehend**

Example: Textual DSL for Architecture Description

```
component DelayCalculator {
      provides aircraft: IAircraftStatus
      provides managementConsole: IManagementConsole
      requires screens[0..n]: IInfoScreen
}
component Manager {
      requires backend[1]: IManagementConsole
}
component InfoScreen {
      provides default: IInfoScreen
}
component AircraftModule {
      requires calculator[1]: IAircraftStatus
}
```

Example: Graphical DSL for J2ME App



USING MODELS AS CENTRAL ARTIFACTS OF THE DEVELOPMENT PROCESS

How can we use models as central artifacts of the development process?

Most common today: Software is in part or entirely **generated** from models

- A code generator is employed to create the software
- Usually not entirely generated: what is used only once, should be hand-coded
- Only the schematic recurring code will be generated

How can we use models as central artifacts of the development process?

Alternative: Models are part of the software and **interpreted or evaluated** at runtime

Many implementations:

- Models used in interpreters e.g. of scripting languages
- Model VMs
- Rule Engines
- Process or Workflow Engines

WHAT IS CODE GENERATION?

Code Generation

In MDD: Transforming models into executable code

Analogy

Input: Water

Transformation

Output: Electricity

Sample Model-driven Tool Chain in SW Development

Constraints

```
# correlation group package must be stereotyped as CorrelationGroup
CorrelationIdentifier addInvariant {
          [FCL forAll cg [self correlationGroup] {
              [FCL exists cgi [FCL allInstances Correlation::CorrelationGroup] {
                    [$cgi basePackage] == $cg
              }]
          }]
}
# each CorrelationGroup must have a CorrelationIdentifier
CorrelationGroup addInvariant {
          [FCL exists cid [FCL allInstances Correlation::CorrelationIdentifier] {
              [$cid correlationGroup] == [self basePackage]
          }]
}
```

DSLs / Modeling Tools

Modeling tool

Texutal DSL code

Model Instances

Model Validator

Code Generator

Generated Code

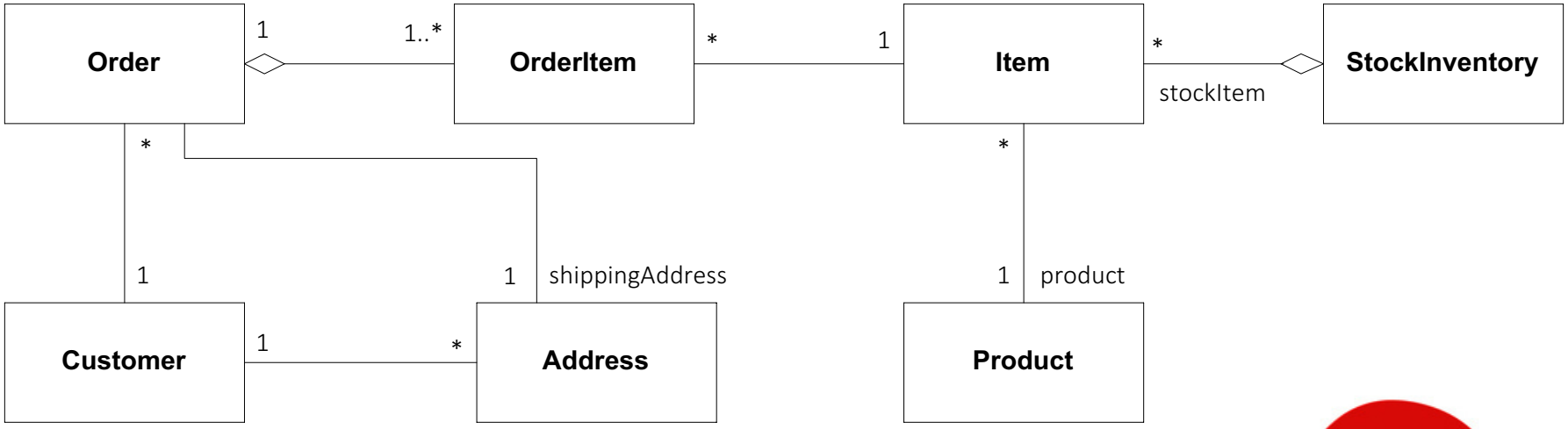
Code

Transformation Templates

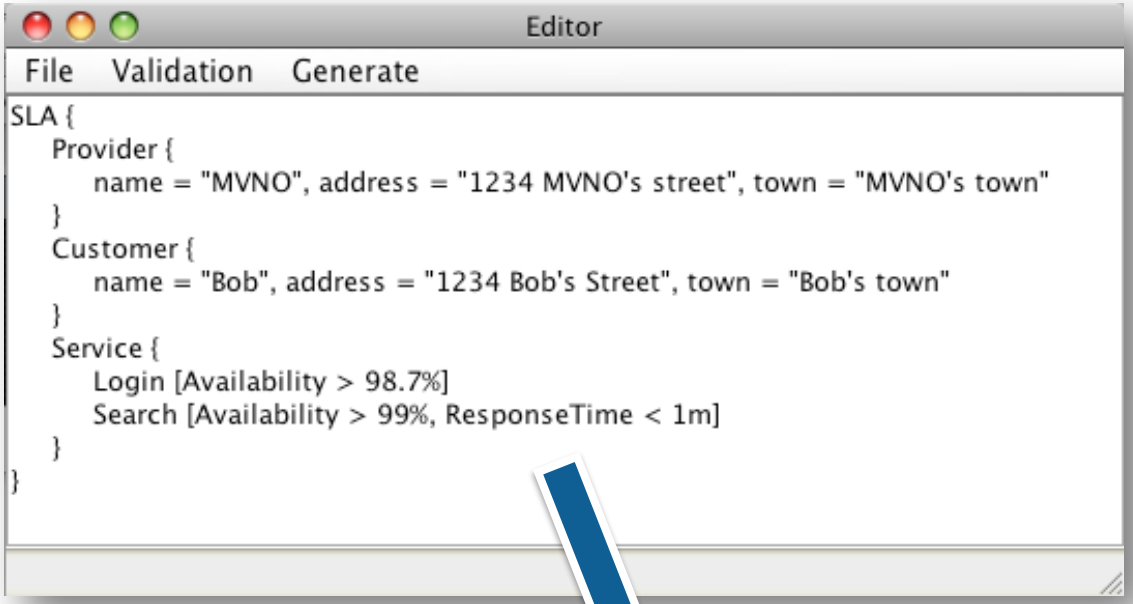
```
#
# Template for the Sequence structure
#
«DEFINE Activity(core::View iv, core::View cv, core::View hcv) FOR orchestration::Sequence»
          <sequence>
                    «EXPAND Activity(iv, cv, hcv) FOREACH element»
          </sequence>
«ENDDEFINE»

#
# Template for the Flow structure
#
«DEFINE Activity(core::View iv, core::View cv, core::View hcv) FOR orchestration::Flow»
          <flow>
                    «EXPAND Activity(iv, cv, hcv) FOREACH element»
          </flow>
«ENDDEFINE»
```

Can we generate code from this model?



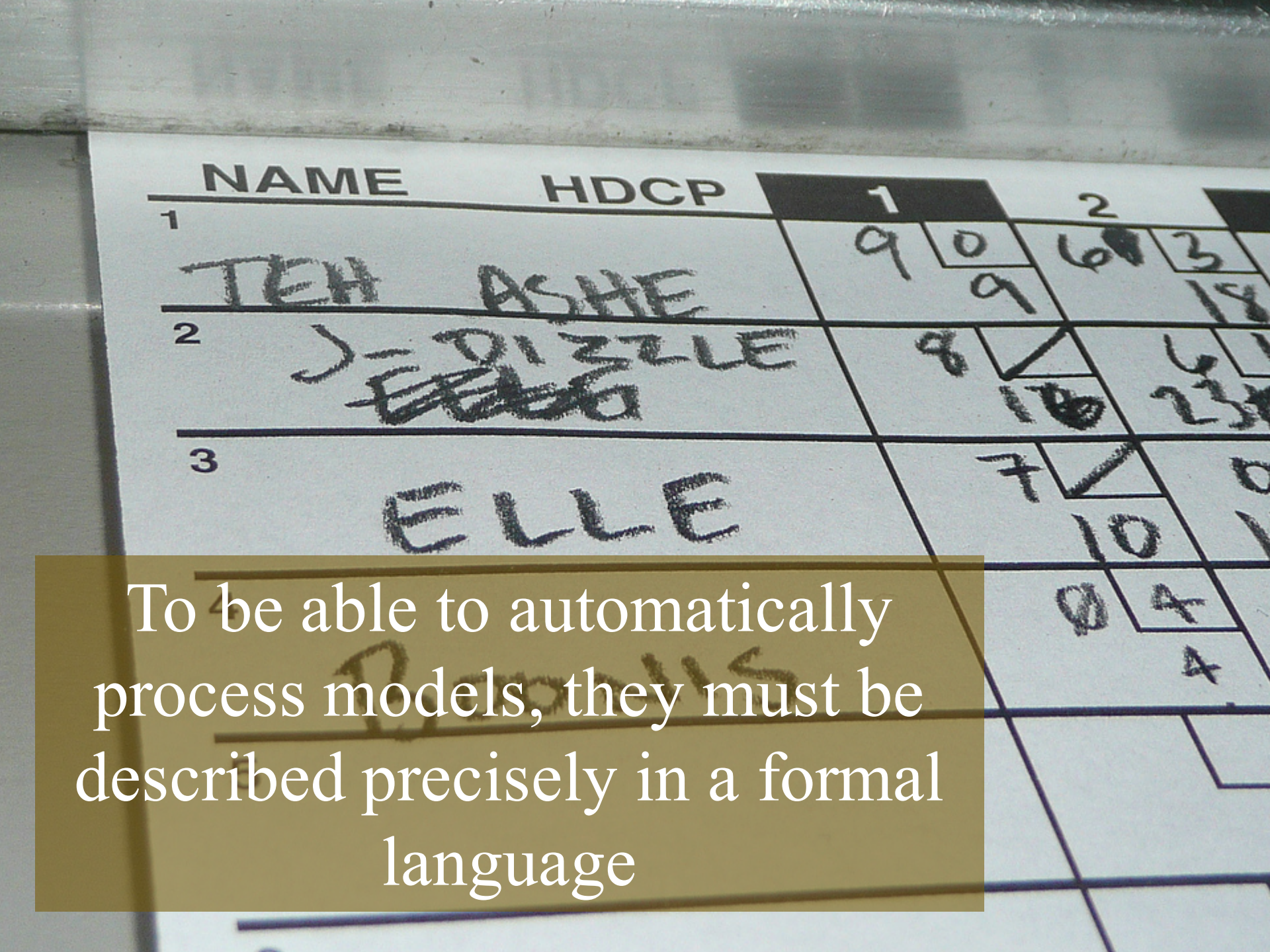
Example: High-Level and Low-Level QoS DSLs



High-Level DSL

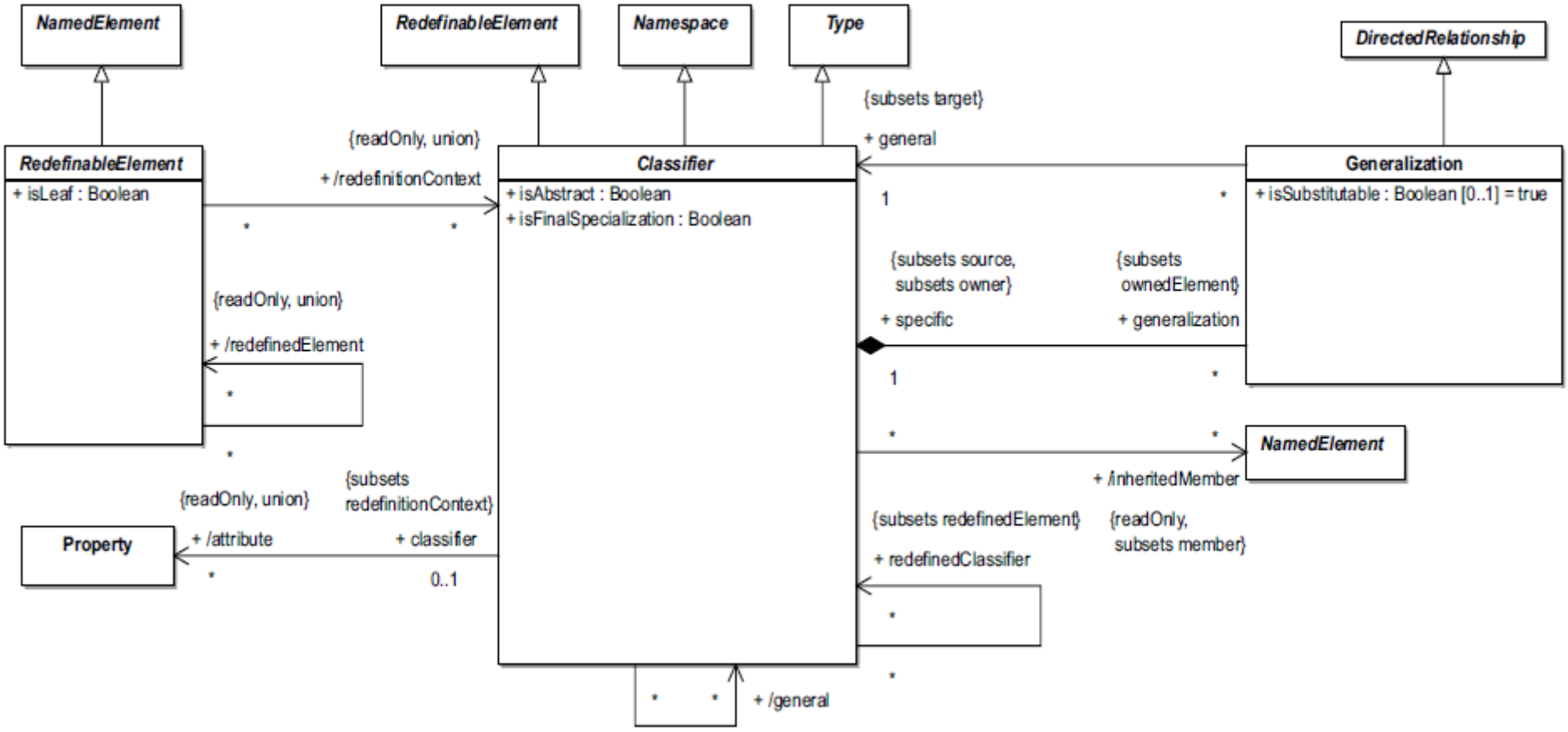
Low-Level DSL

```
## SEARCH service's technical concerns
SearchService classes cxf::Service
SearchService package "mvno"
SearchService uri "http://localhost:9000/watchme/search"
SearchService operations [Operation create search -name "search" \
    -parameters [list build
    [Parameter create movietitle -set name "movietitle" -set type String]
    [Parameter create movietitle -set name "language" -set type String]]]
```

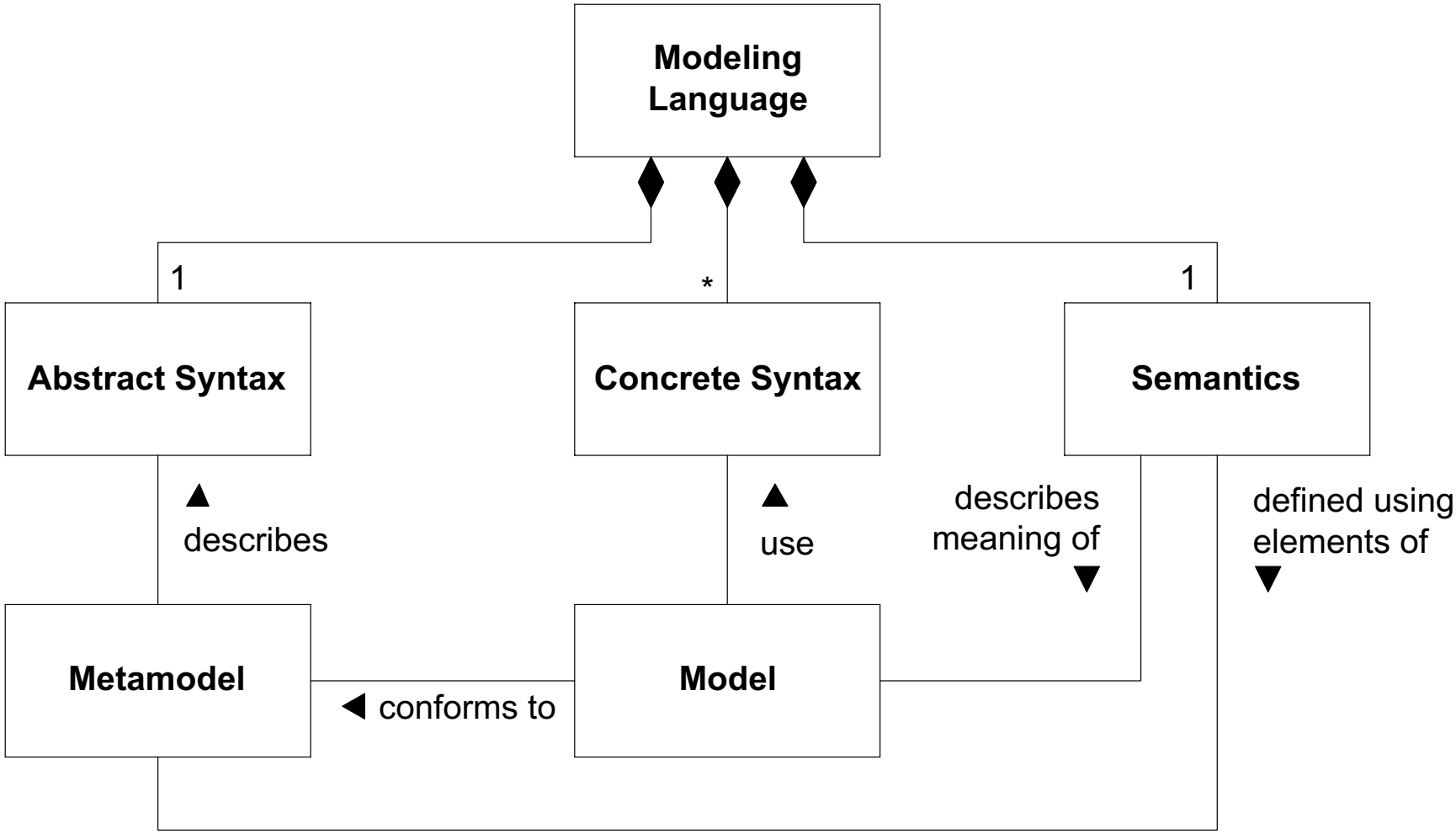


To be able to automatically process models, they must be described precisely in a formal language

Example: Classifiers in the UML2 Meta-model



Relationships of Modeling Language and Meta-Model



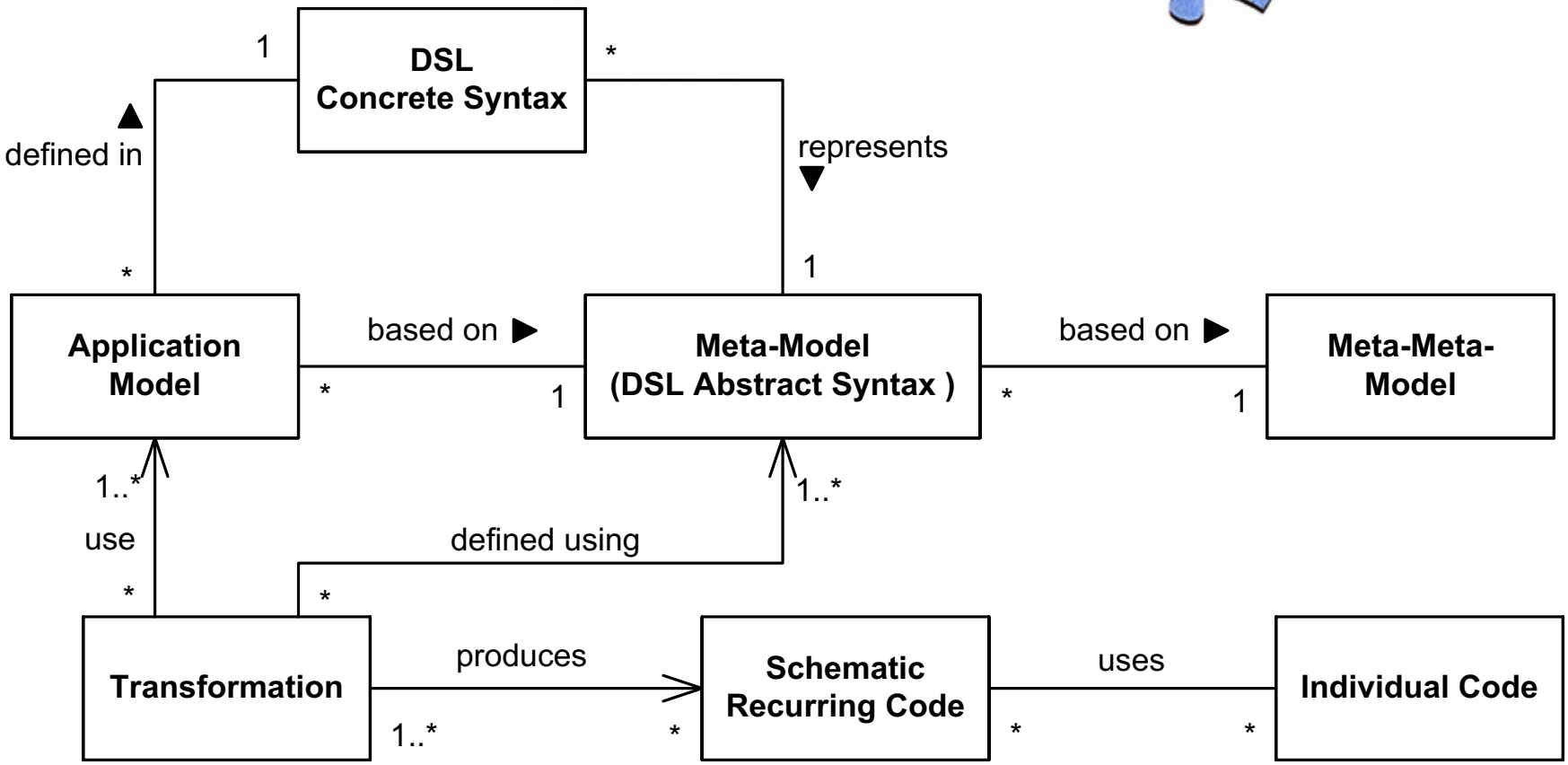
There is no silver bullet

MDD requires to **create an infrastructure** consisting of modeling tools, generators, platforms, etc.

Much effort must be spent for **domain analysis**.

These **efforts do usually not pay off for using the MDD infrastructure only once.**

Putting it all together



Relevant Literature and Sources

- Markus Völter. Modellgetriebene Software-Entwicklung. http://www.voelter.de/data/articles/MDSD.pdf
- Tom Stahl, Markus Völter, Sven Efftinge, Arno Haase. Modellgetriebene Softwareentwicklung, 2. Auflage. Techniken, Engineering, Management. dPunkt, Mai 2007.
- Markus Voelter, Tom Stahl: Model-Driven Software Development. Wiley, 2006.

Many thanks for your attention!

Uwe Zdun

Software Architecture Research Group
Faculty of Computer Science
University of Vienna
http://cs.univie.ac.at/swa