IEEE Ninth International Conference on Research Challenges in Information Science, May 13-15 2015, Athens, Greece

A Domain-specific Language for Modeling Method Definition: From Requirements to Grammar

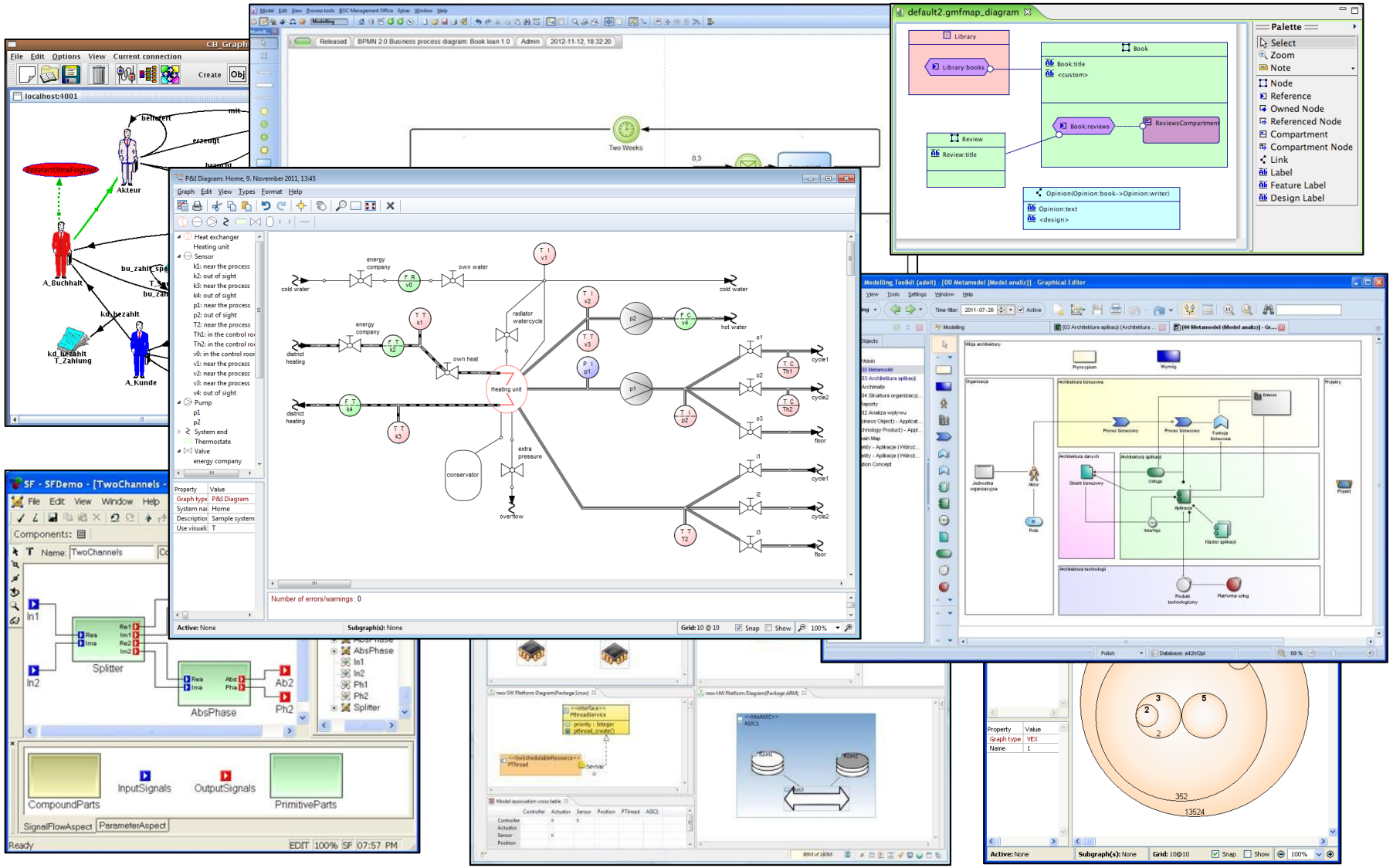
Niksa Visic, Hans-Georg Fill, Robert Buchmann, Dimitris Karagiannis

Agenda

- Introduction and Motivation
- Requirements for a DSL for Modeling Method Definition
- Grammar of the DSL
- Evaluation of the Feasibility of the Grammar
- Conclusion and Outlook

INTRODUCTION & MOTIVATION

Conceptual Modeling Tools

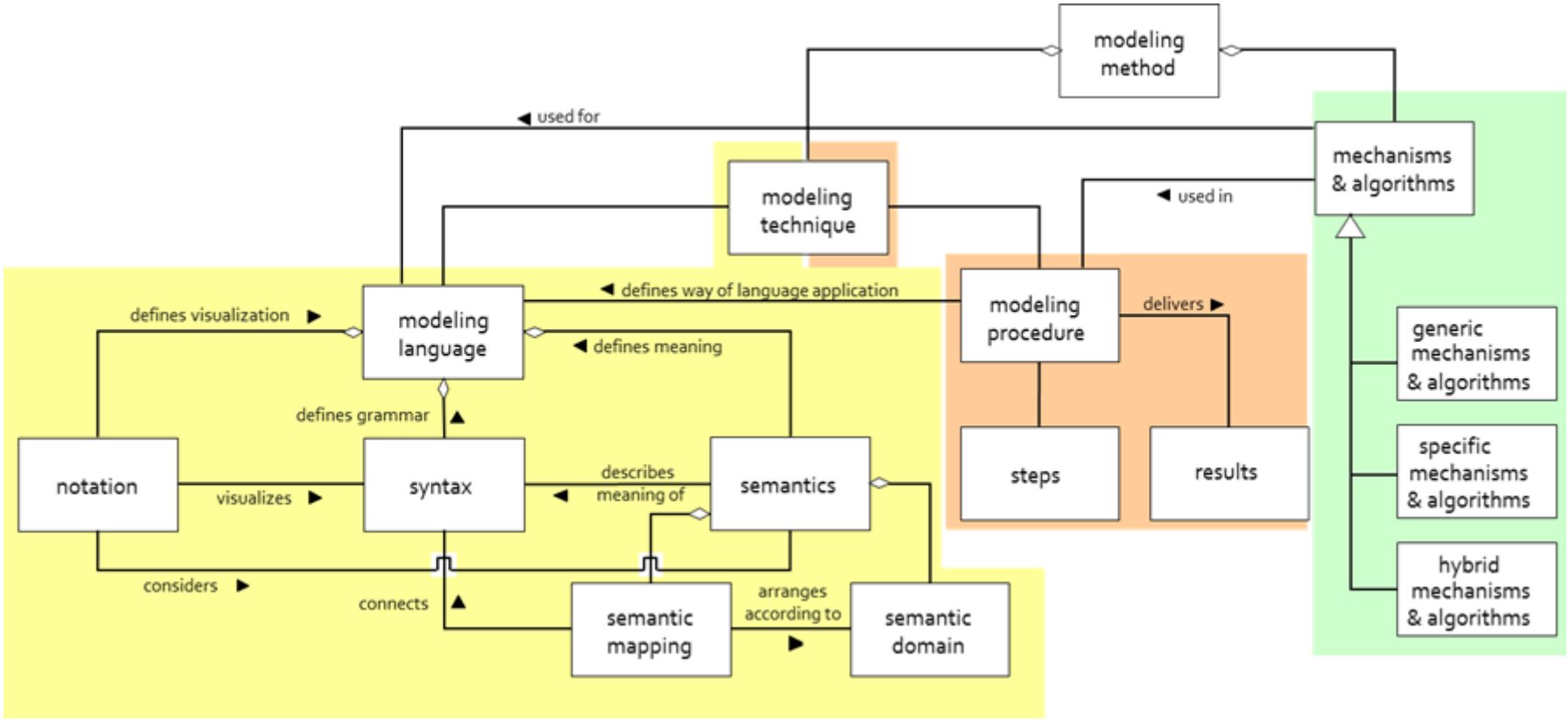


universität wien

Niksa Visic, Hans-Georg Fill, Robert Buchmann, Dimitris Karagiannis

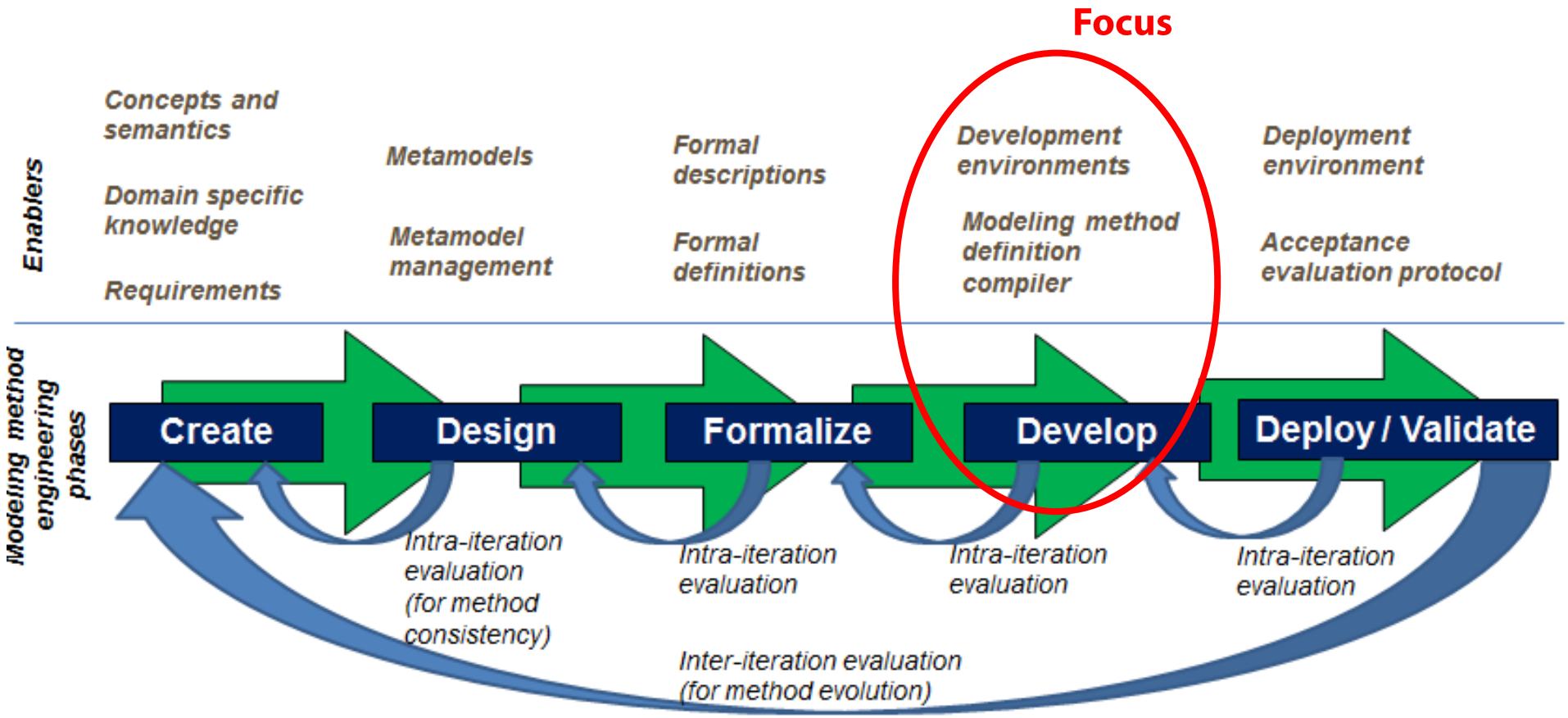
OMiLAB®
www.omilab.org

Components of Modeling Methods



Karagiannis & Kühn (2002)

Agile Modeling Method Engineering

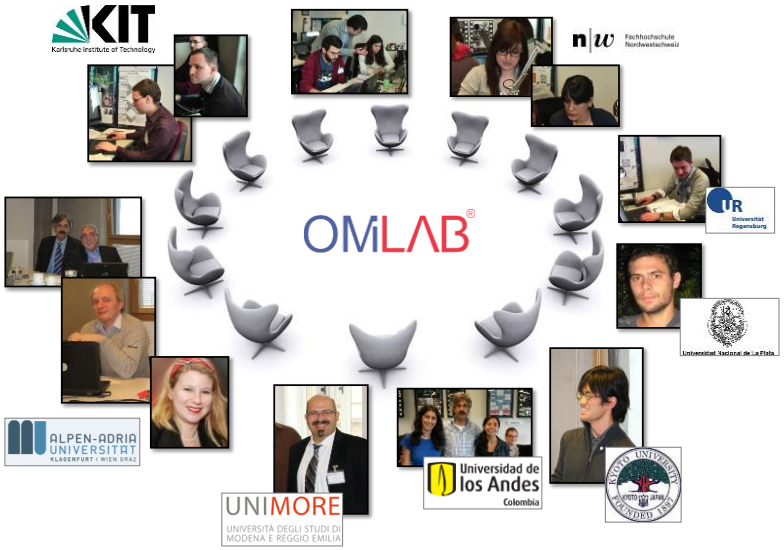


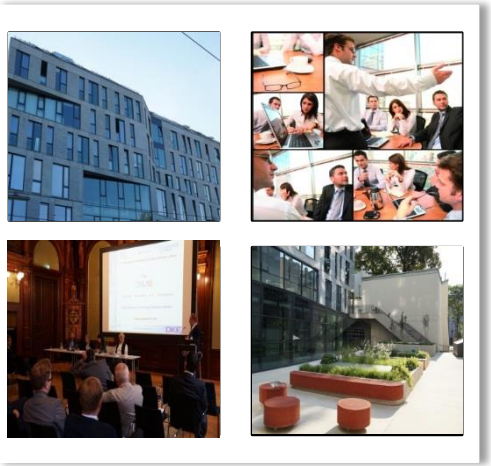
Modeling Method Engineering @ OMiLAB, Vienna, AT

OMiLAB Approach – **Visit www.omilab.org**

- A research and experimental laboratory for the conceptualization, development and deployment of modeling methods and the models designed with them.
- Project space for engineering of modelling methods and modeling tools
- A space for a community of researchers and practitioners sharing a common understanding about model value

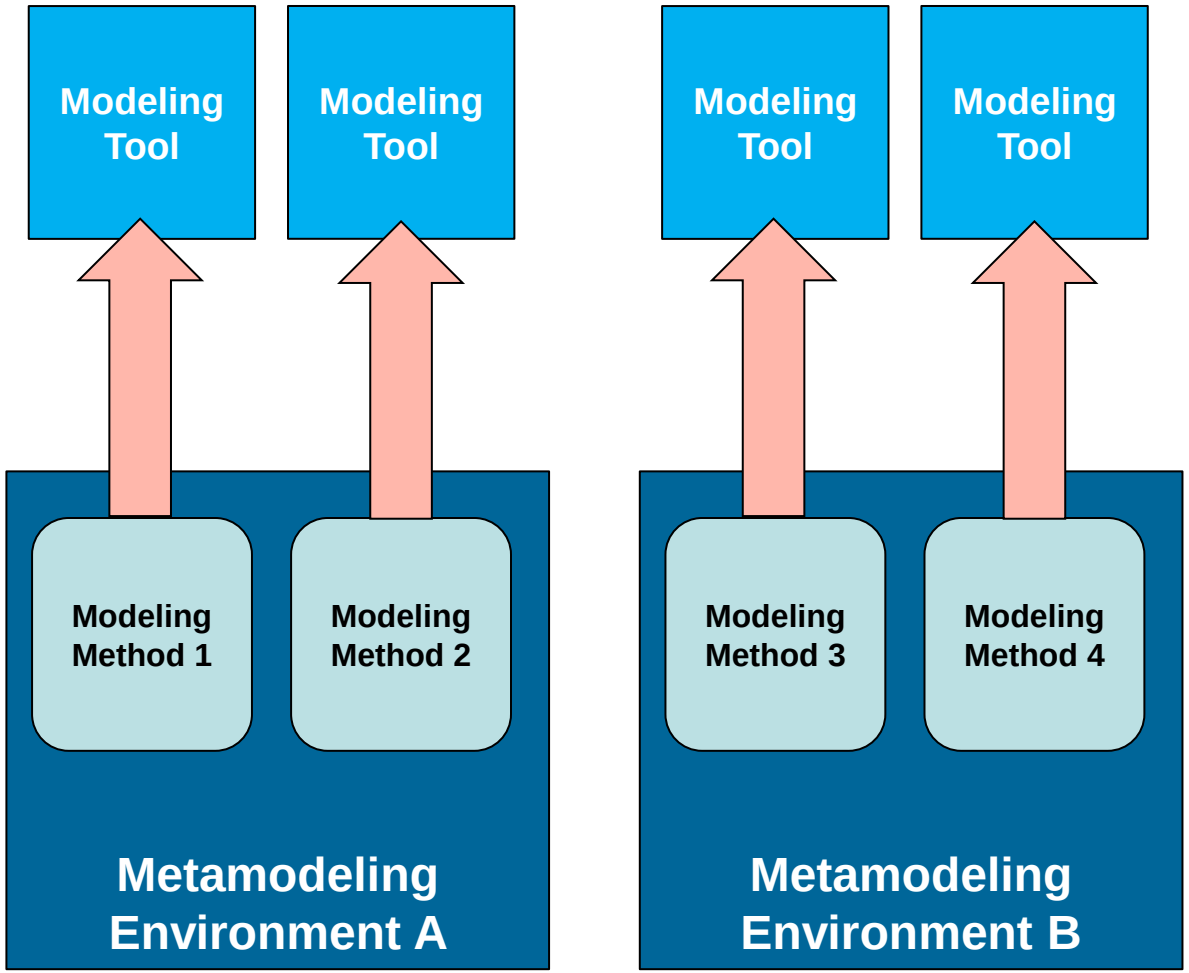
Recent OMiLAB Visitors





OMiLAB@Faculty of Computer Science
Währinger Str. 29, 1090 Vienna, Austria

Realization Using Metamodeling Environments



www.adoxx.org

MetaEdit+

Obeo Designer

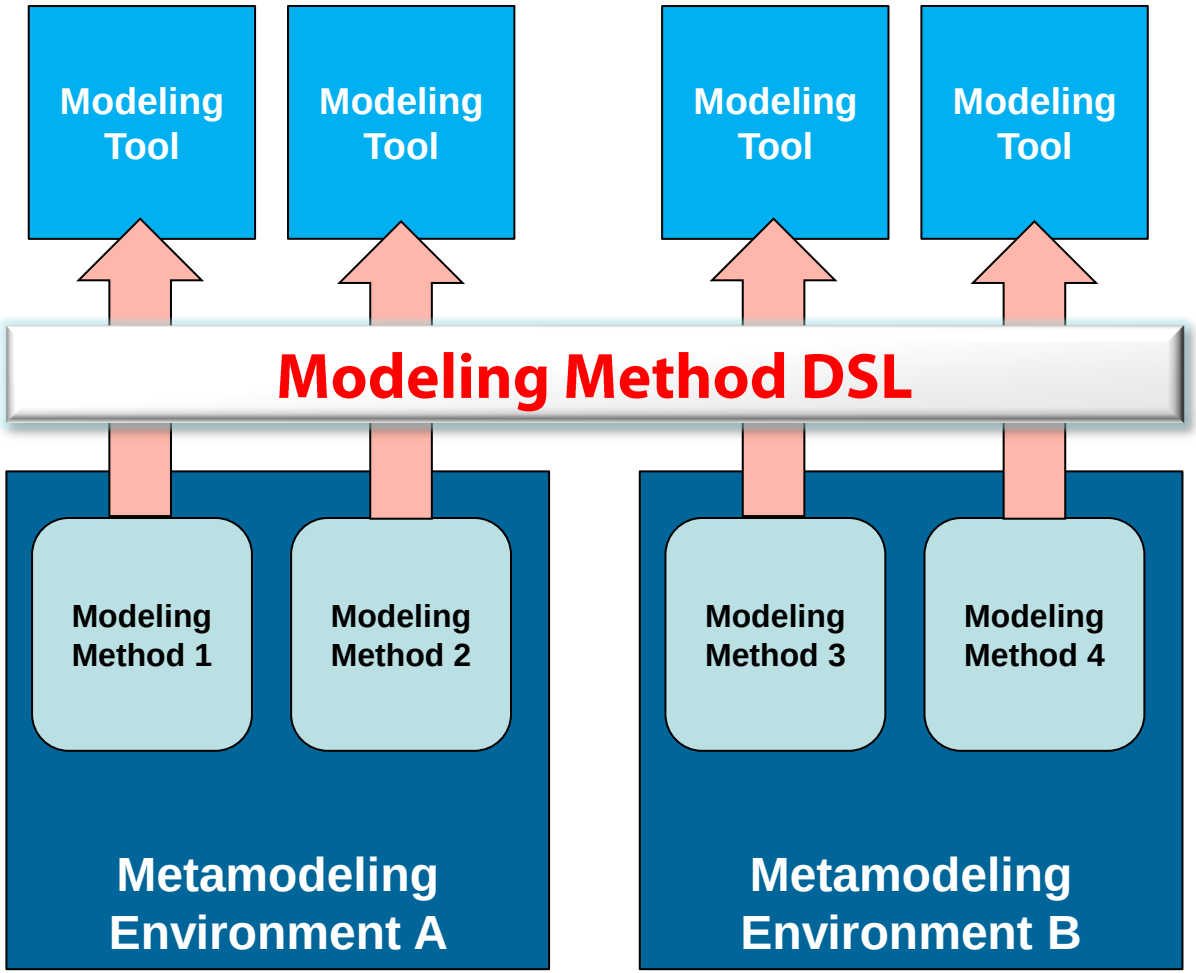
emf
ECLIPSE MODELING FRAMEWORK

Microsoft Visual Studio

GME 11

ConceptBase.cc

Goal: Abstract from Metamodeling Environments



www.adoxx.org

MetaEdit+

Obeo Designer

emf
ECLIPSE MODELING FRAMEWORK

Microsoft Visual Studio

GME 11

ConceptBase.cc

REQUIREMENTS FOR A DSL FOR MODELING METHOD DEFINITION

universität wien

Niksa Visic, Hans-Georg Fill, Robert Buchmann, Dimitris Karagiannis

OMiLAB®

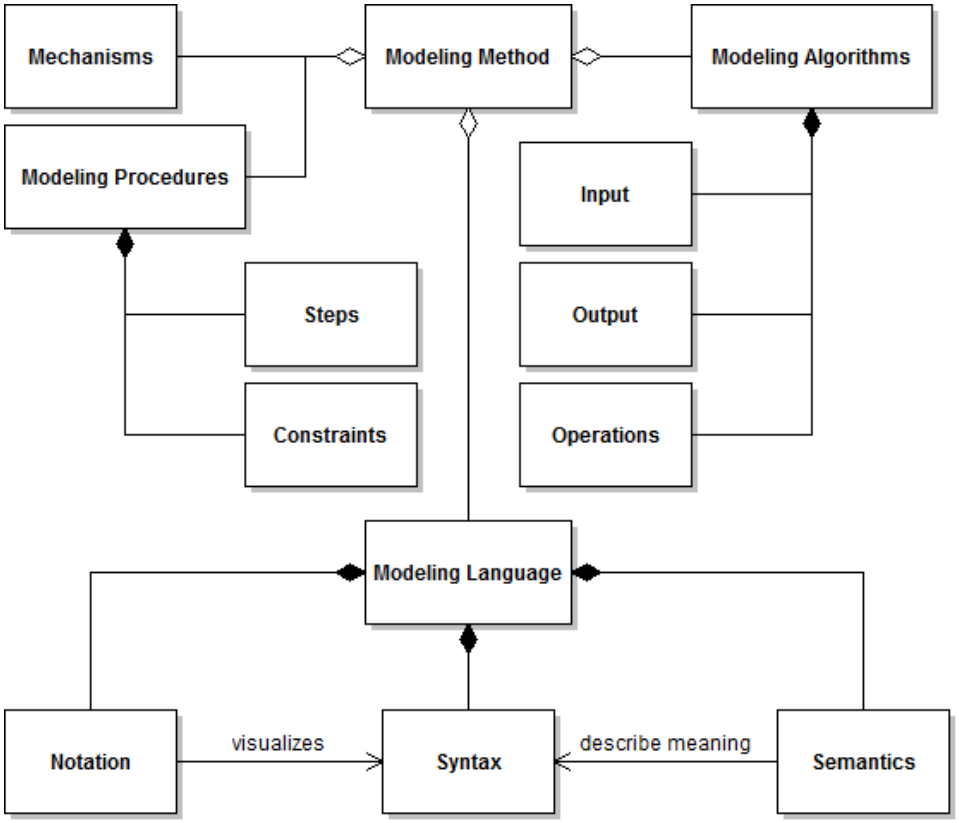
www.omilab.org

The Six Major Requirements for MM-DSL

A. Application Domain: Modeling Method Engineering
B. Analyzing the Concepts from the Application Domain
C. Analyzing Existing Artefacts from the Domain
D. Analyzing Meta-modeling Platform Functionality
E. Analyzing Best Practices and Guidelines for the Design of DSLs
F. The Evolution Requirement

Analysis of Application Domain

Application Domain: Modeling Method Engineering



Refinement for DSL based on Karagiannis & Kühn (2002)

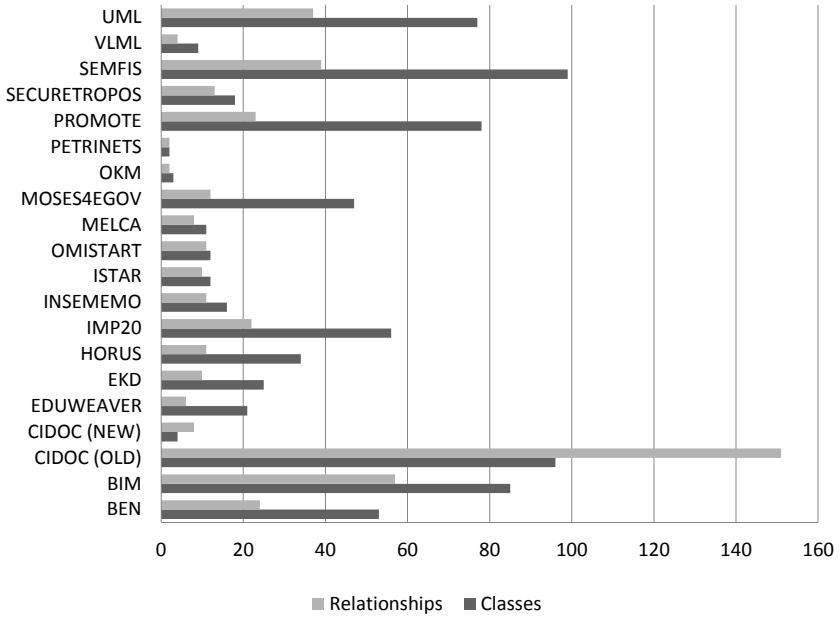
Analysis of Domain Concepts

Analysis of Domain Concepts:

- Investigation of existing modeling methods such as BPMN, Petri Nets, UML
- Analysis of primitive elements in existing meta^2 models, e.g. ADOxx, GME, GOPPR, …
- Derivation of sets of meta modeling concepts, e.g. class, relationship, attribute, …
- Analysis of relationships and functionalities of meta^2 models, e.g. domain, range, specialization, decomposition, model/diagram types,…
- Analysis of constraint and rule specification options
- Analysis of building blocks for describing algorithms

Analysis of Existing Artefacts

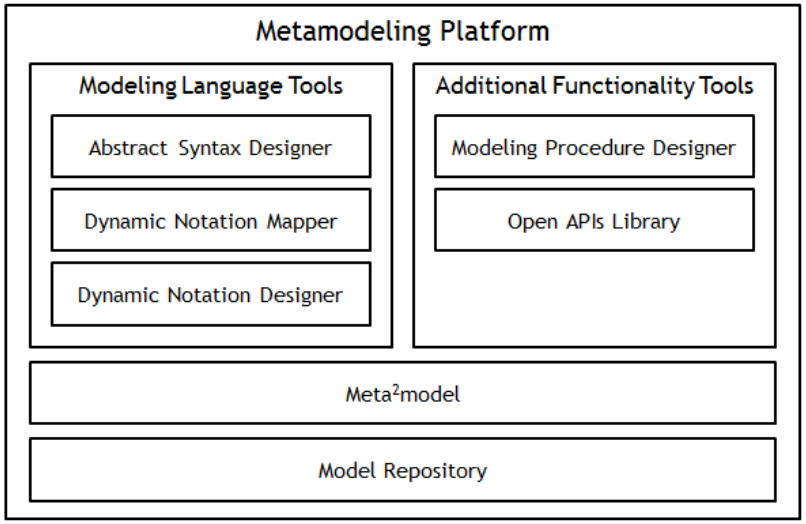
- Analysis of 19 implementations of modeling methods developed within OMiLAB: BEN, BIM, eduWEAVER, EKD, HORUS, IMP2.0, i*, OMi*T, InSeMeMo, MeLCa, OKM, Secure Tropos, UML, PetriNets, MoSeS4eGov, PROMOTE, SeMFIS, and VLML



Classes		Relationships	
Name	***%***	***Name***	***%***
Container	68	Associates	42
Actor	53	Depends	37
Label	53	Flows	37
Resource	53	Specializes	37
Decision	47	Has	32
Activity	37	Part of	32
Process	37	Contributes	21
Start	37	Decomposes	21
End	37	Relates	21
Goal	32	Uses	16

Analysis of Meta-modeling Platform Functionality

- Genericity of meta^2 models
- Structured abstract syntax and semantics definition
- Mapping mechanisms between abstract syntax and graphical notation
- UI embedded in modeling elements
- Enforcement of modeling procedures
- APIs for algorithm development
- Repository for storing modeling method definitions and models
- Extensibility, Interoperability, Scalability



Best Practices and Guidelines for the Design of DSLs

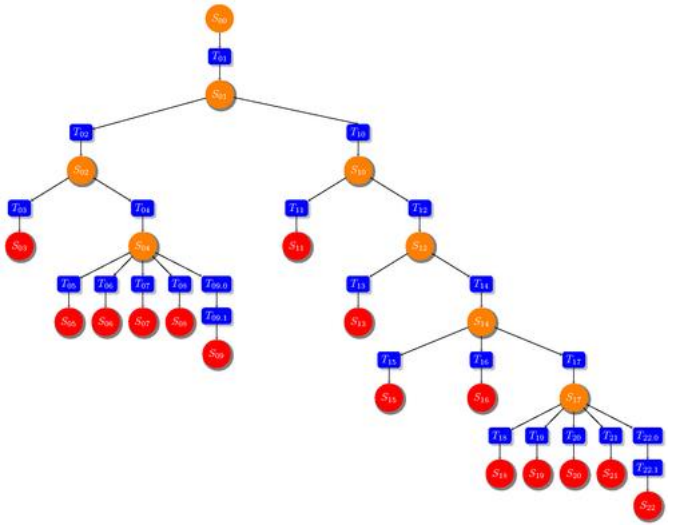
Desirable features:

- ❖ User expectation conformity
- ❖ Readable and consistent syntax
- ❖ Small and orthogonal set of features
- ❖ Error diagnosis

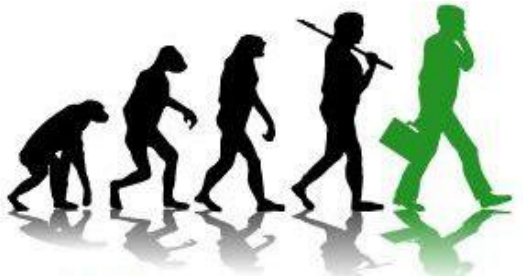
```
public void processData()
{
  do
  {
    int data = getData();
    if(data < 0)
      performOperation1(data);
    else
      performOperation2(data);
  }
  while(hasMoreData());
}
```

Undesirable features:

- ○ Paradigmatic purity
- ○ Language bloat
- ○ Syntactic synonyms
- ○ Syntactic homonyms
- ○ Hardware dependency
- ○ Backward compatibility



Evolution Requirement



Potential evolution of DSL due to:

- Future development of meta-modeling platforms
- Possible changes in the application domain

Requires:

- Extensible abstract syntax, concrete syntax and semantics
- Extensible execution engine

GRAMMAR OF THE DSL

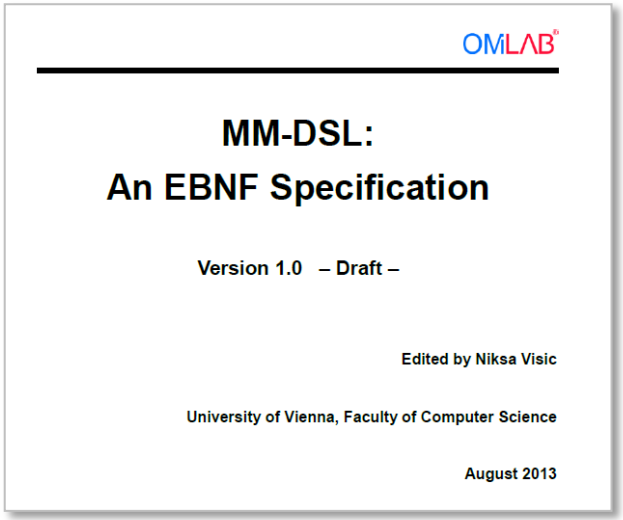
Description of the DSL Grammar

- Design decision for a simple declarative style
- Easy realization using XText and Xbase

Statement	Statement Specification in EBNF	Meaning
Root	root ::= methodname embedcode* method	The root of the method definition document
Method Name	methodname ::= 'method' name	
Embed	embedcode ::= 'embed' name '<' name-embedplatformtype (':' name-embedcodetype)? '>' 'start' embeddedcodegoeshere 'end'	In the case that native code for the target metamodeling platform will be embedded, the platform must be declared.
Method	method ::= enumeration* symbolstyle* symbolclass* symbolrelation* metamodel algorithm* event*	Container for the method building blocks and auxiliary elements
Enumeration	enumeration ::= 'enum' name '{' enumvalues+ '}'	An auxiliary element of a method definition, defining a list of values (typically used to restrict attributes of modeling objects)
Metamodel	metamodel ::= class+ relation* attribute* modeltype+	The main building block of a method, describing structurally the language metamodel
Class	class ::= 'class' name ('extends' name-class)? ('symbol' symbolclass)? '{' (attribute | insertembedcode)* '}'	The definition of a modeling concept, including assignment of its graphical notation (if instantiable), its editable property set and prescribed inheritance, to be instantiated by modeling objects
Relation	relation ::= 'relation' name ('extends' name-relation)? ('symbol' name-symbolrelation)? 'from' name-class 'to' name-class '{' (attribute | insertembedcode)* '}'	The definition of a modeling relation, including assignment of its graphical notation, its editable property set and prescribed inheritance
Attribute	attribute ::= 'attribute' name ':' type ('access' ':' acesstype)?	The definition of a property (for a modeling concept or relation)

Public Availability and Extension of DSL Grammar

- Grammar of the DSL grammar publicly available via www.omilab.org
- Formal specification through EBNF
- Ready for custom-developed extension
- Feedback and requests to grammar are highly welcome


OMLAB

MM-DSL:
An EBNF Specification

Version 1.0 – Draft –

Edited by Niksa Visic

University of Vienna, Faculty of Computer Science

August 2013

Example for Using the Grammar

```
// the smallest method possible
method SmallestInstantiable

class Abc {}

modeltype ModelTypeA
{
 classes Abc
 relations none
 modes none
}
```

```
style RedBlue { fill:red stroke:blue stroke-width:4 }

classgraph MotorGraph style RedBlue {
 rectangle x=-30 y=-30 w = 60 h = 60
 circle cx = 0 cy = 0 r = 30
 ellipse cx = 0 cy = 0 rx = 10 ry = 30
}

class Motor extends Vehicle symbol MotorGraph {
 attribute biggerFrontTire:enum YesNo access:read
 attribute brand:string access:read
}
```

```
algorithm MyAlgorithm {
 ui.item.menu.insert MyAlg to MyMenu
 ui.infobox "MM-DSL Info-Box" "Generated with MM-DSL"
}
```

The smallest possible method that can be translated into a fully functional modeling tool contains only one class and one model type

If graphical notation is not defined, it will be assigned automatically using the symbols from a default pool of symbols

Defining graphical notation is similar to writing SVG code

Graphical symbols and styles can be defined once and referenced infinite amount of times

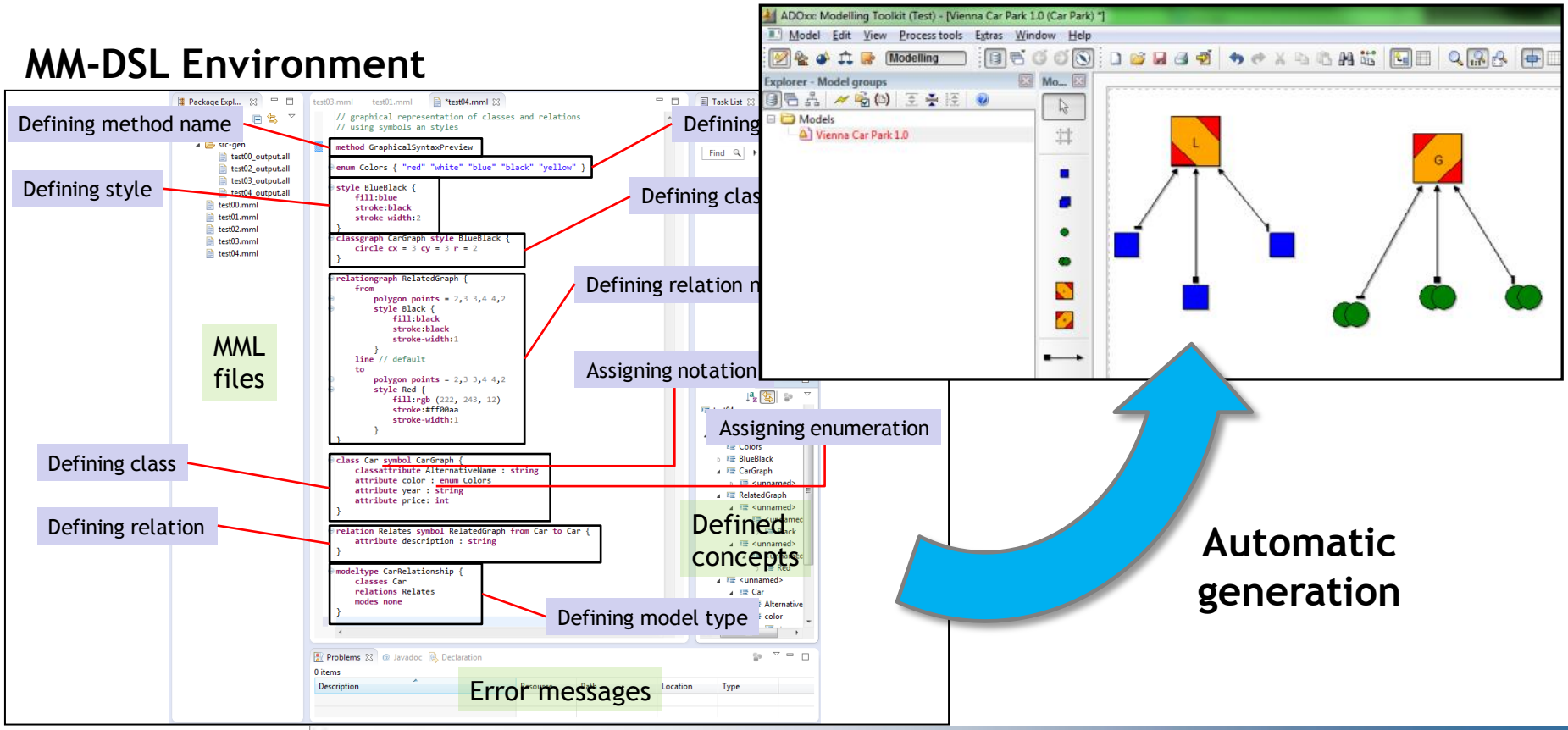
Support for generic algorithms and events is included (creating models or objects, events that trigger algorithms when something changes inside a modeling tool, etc.)

Practical Evaluation

- Prototypical implementation of a modeling method definition environment based on Xtext
- Compiler for the ADOxx meta-modeling platform

ADOxx Modeling Tool

MM-DSL Environment

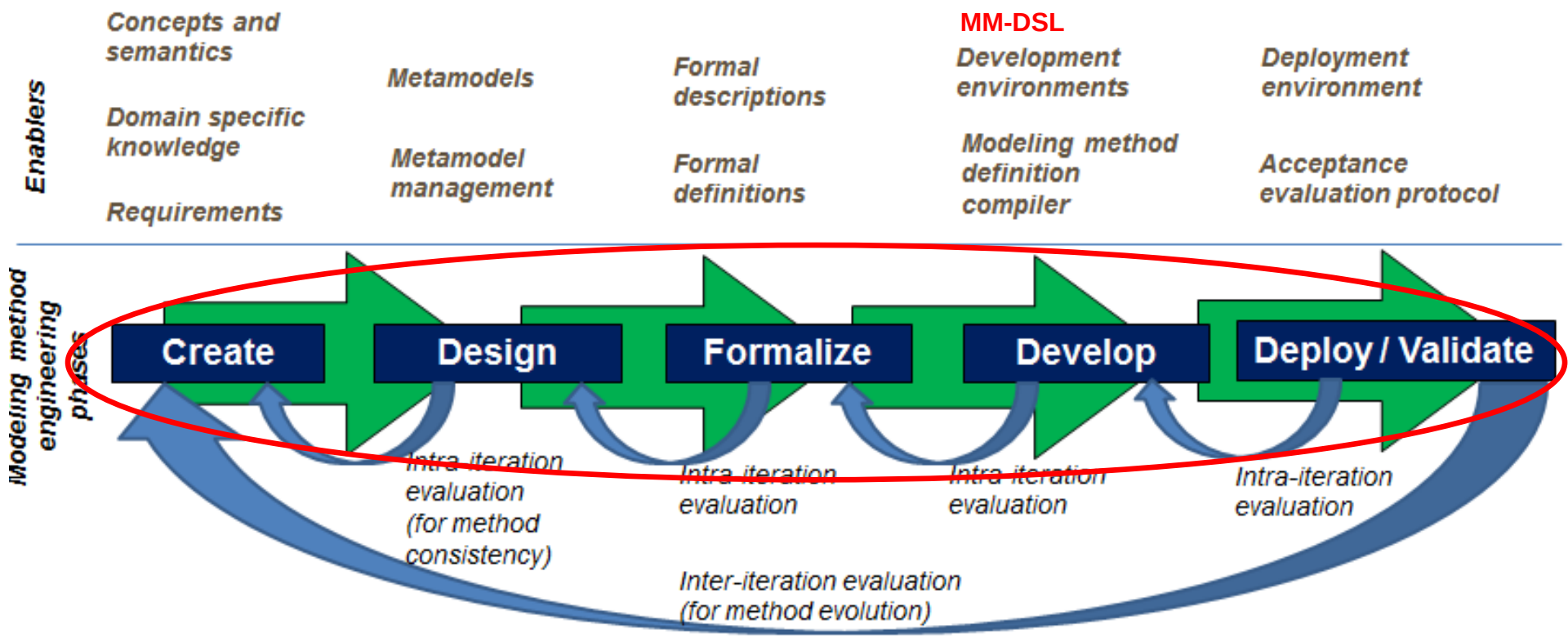


CONCLUSION AND OUTLOOK

Conclusion and Outlook

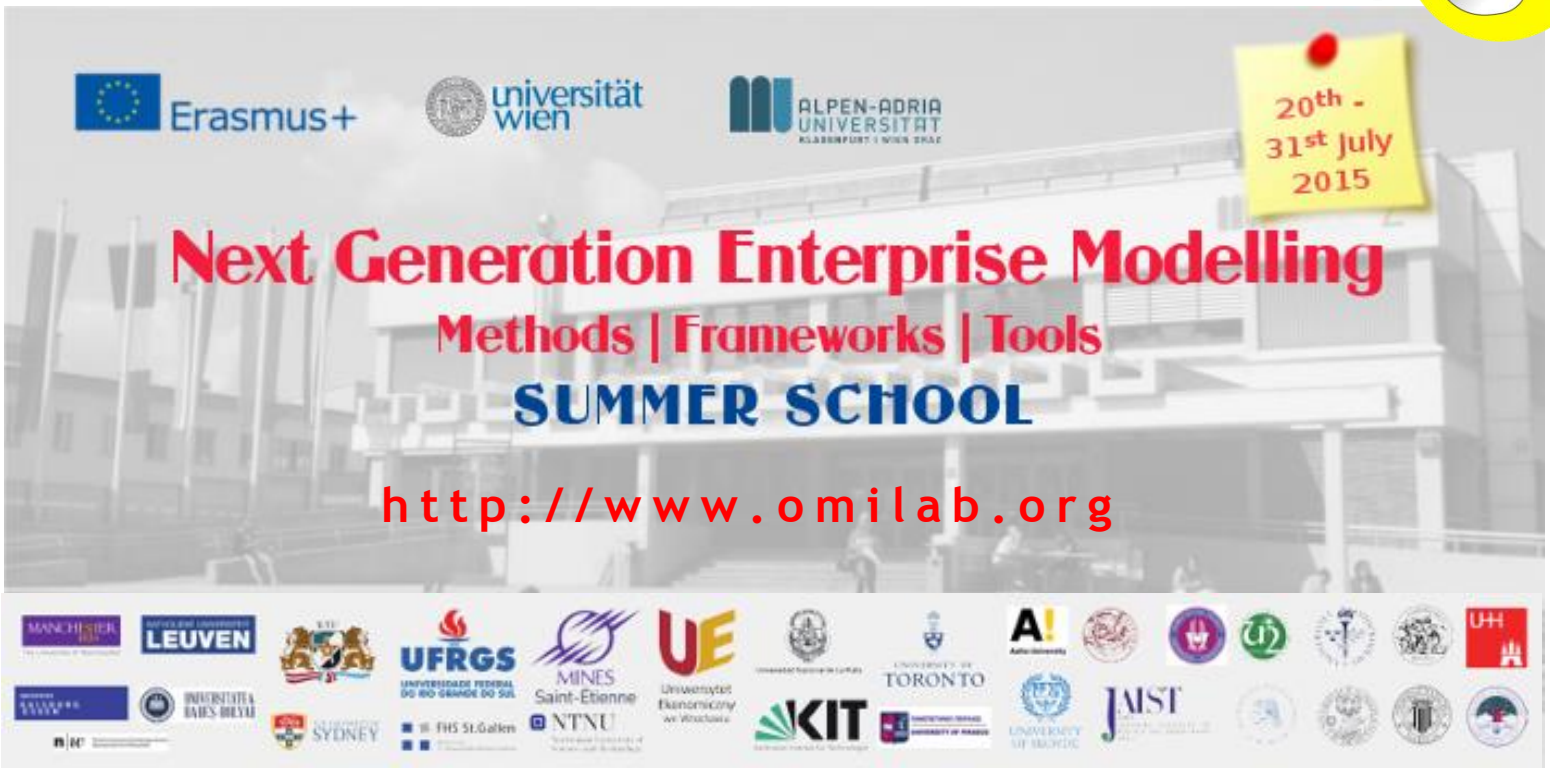
- Approach enables code-based platform-independent definition of conceptual modeling methods
- Compilers necessary to deploy the modeling method definition on meta-modeling platforms
- Potentially applicable to a large number of meta-modeling platforms
- Abstraction layer requires its own learning curve
- Concept of modeling method should be platform-independent
- Further evaluations with other platforms than ADOxx

Outlook: Support all Phases in Agile Modeling Method Engineering



NEMO Summer School 2014-2017

- Summerschools on Next Generation Enterprise Modeling
- Location: Austria (Vienna / Klagenfurt)
- Deadline for 2015 is over
- Please apply next year!

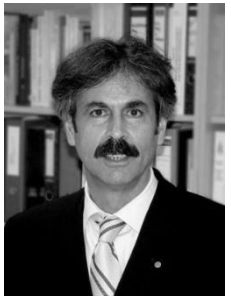


Thank You For Your Attention!

Niksa Visic

Hans-Georg Fill

Robert Buchmann



Dimitris Karagiannis

Selected References

- D. Karagiannis and H. Kühn, “Metamodelling Platforms,” in E-Commerce and Web Technologies, vol. 2455, K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 182.
- H.-G. Fill and D. Karagiannis, "On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform", Enterprise Modelling and Information Systems Architectures, Vol. 8, Issue 1, 2013, pp. 4-25.
- H.-G. Fill, T. Redmond, D. Karagiannis, "FDMM: A Formalism for Describing ADOxx Meta Models and Models", in: L. Maciaszek, A. Cuzzocrea and J. Cordeiro: Proceedings of ICEIS 2012 - 14th International Conference on Enterprise Information Systems, Vol.3, SciTePress, 2011, pp. 133-144.
- “The ADOxx meta-modeling platform" [Online]. Available: http://www.adoxx.org/.
- D. Karagiannis and N. Visic, “Next Generation of Modelling Platforms,” in Perspectives in Business Informatics Research, vol. 90, J. Grabis and M. Kirikova, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 19–28.