
# Next Generation of Modelling Platforms

Dimitris Karagiannis

Niksa Visic

University of Vienna, Faculty of Computer Science,

Research Group Knowledge Engineering

universität wien

Was founded by **Duke Rudolph IV in 1365**. It is the oldest University in the German-speaking cultural area and one of the largest in Central Europe.

The University of Vienna is the largest teaching and Research institution in Austria, with ca. 6,200 persons academic staff. It aims to sustain a wide range of studies as well as to promote new and innovative fields of research.

Currently, about 72,000 students are enrolled in more than 130 courses, of which 34 are Diploma Programmes, 26 Bachelor Programmes and 46 Master Programmes.

# Business Informatics at the universität wien

- Business Informatics research supposed to be beneficial for society and business, based primary on !
  - Behavioristic research
  - Design-oriented research
- Most prominent objective:
  - To position design-oriented IS research in the international research community.
  - Produce practically beneficial, business relevant results.

**Memorandum on Design-Oriented Information System Research:**
**www.dke.univie.ac.at**

Hubert Österle, Jörg Becker, Ulrich Frank, Thomas Hess, Dimitris Karagiannis, Helmut Krcmar, Peter Loos, Peter Mertens, Andreas Oberweis and Elmar J. Sinz

# Why Model ?!

REVEAL THE APPARENTLY SIMPLE (COMPLEX)
TO BE COMPLEX (SIMPLE)

DESIGN AND REDESIGN

SUGGEST EFFICIENCIES

DISCOVER NEW QUESTIONS

ANALYZE AND SIMULATE

DEMONSTRATE TRADEOFFS

PREDICTION

DOCUMENTATION

OPTIMIZE

ILLUMINATE UNCERTAINTIES

EXECUTION

DATA COLLECTION

EXPLAIN

**Modelling as Horizontal Function!**

**- Covering all domains of Computer Science**

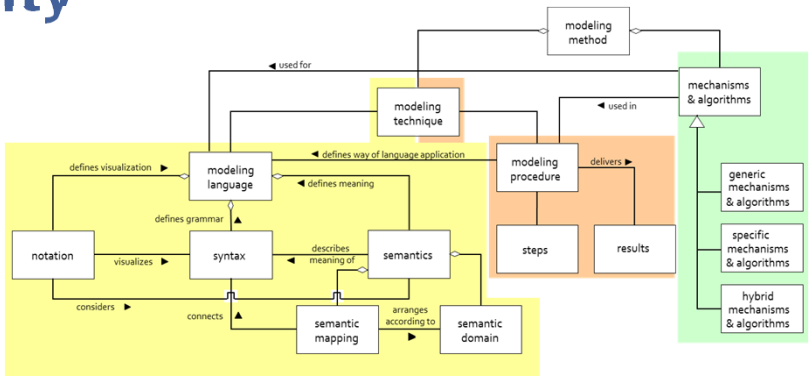
# Agenda: The Challenge

What do we like to support?

The „conceptualazation of modelling methods" process

# How the Scientific Community uses the Terms…

- Modelling Languages
- Modelling Methods

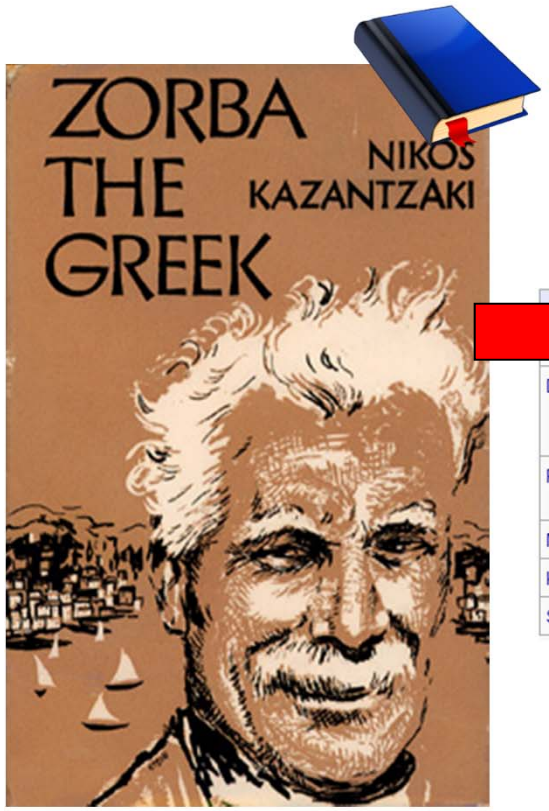



- Sometimes these terms are used “synonymously”

**Modelling Method ≥ Modelling Language**

- Note: Modelling language is one of necessary parts of a modelling method.

# “From Book to Movie”: A Metaphor




| / Staff | |
|---|---|
| | Cacoyannis |
| Drehbuch: | Michael Cacoyannis ( dem Roman von Niko Kazantzakis) |
| Produktion: | Michael Cacoyannis, Anthony Quinn |
| Musik: | Mikis Theodorakis |
| Kamera: | Walter Lassally |
| Schnitt: | Michael Cacoyannis |

| Filmdaten / Film Data | |
|---|---|
| Deutscher Titel: | **Alexis Sorbas** |
| Originaltitel: | **Zorba the Greek** |
| Produktionsland: | Vereinigte Staaten, Vereinigtes Königreich, Griechenland |
| Erscheinungsjahr: | 1964 |
| Länge: | 142 Minuten |
| Originalsprache: | Englisch |
| Altersfreigabe: | FSK 12 |

**Besetzung / Cast**

- Anthony Quinn: Alexis Sorbas
- Alan Bates: Basil
- Irene Papas: Witwe
- Lila Kedrova: Madame Hortense
- Sotiris Moustakas: Mimithos
- Yorgo Voyagis: Pavlis

# “From Book to Movie”: Directed by is like …

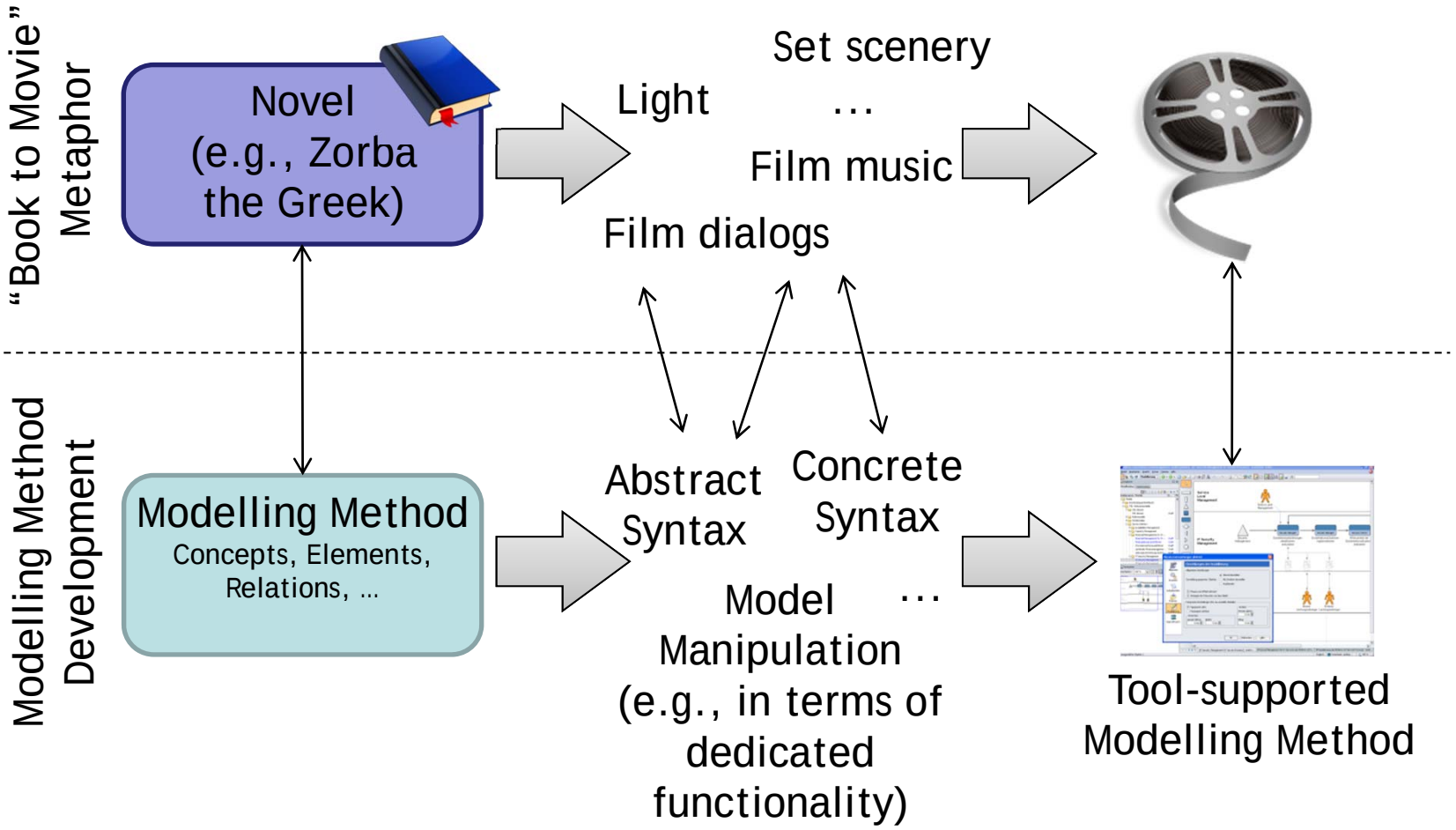
**Screen Play**

- Dialogs
- Interactions
- Scenery
- Effects
- …

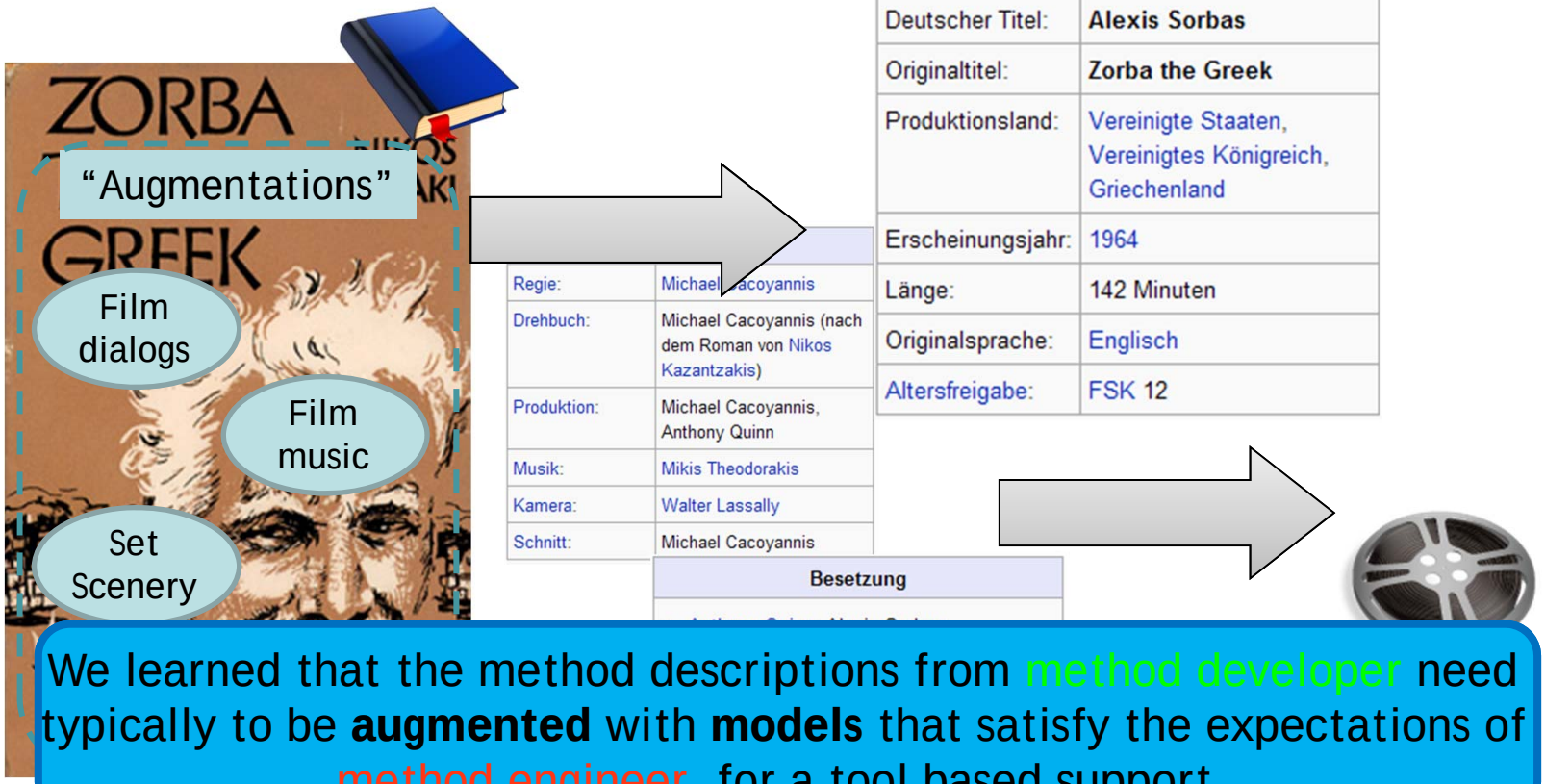
**Writing** → **Acting** → **Cutting**

## What is the Analogue for Modelling Methods ?

# … Continue the Metaphor



# The Method Conceptualization process




We learned that the method descriptions from method developer need typically to be **augmented** with **models** that satisfy the expectations of method engineer for a tool based support.

We named this process: **The Method Conceptualization process**

# The Method Conceptualization process

- Capturing of fundamental **concepts**, **relationships** in between and **properties** adhering to them, usually obtained through the analysis of a selected domain
- Descriptions of such conceptualizations varies depending on the **addressed audience**, with different expectations
  - End User, Modeler, Developer, ….
- From a development perspective, a method conceptualization needs to be **formal enough** to enable developer continue along the life-cycle
  - A **model** of the method (language) that facilitates a **coherent view** on the core concepts involved

# ..... guide to a modelling method Tool

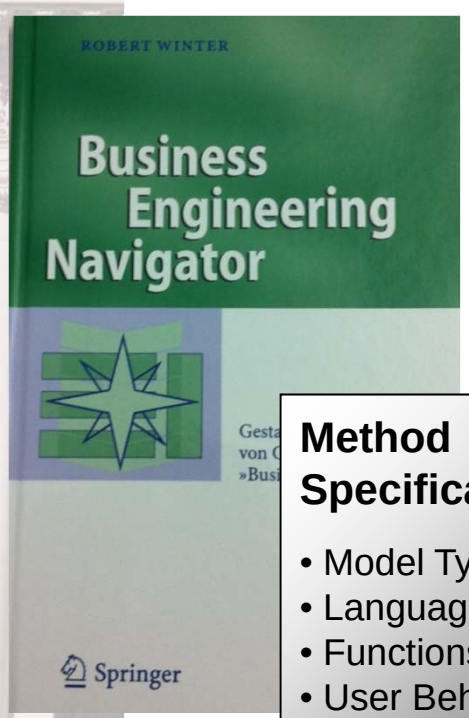
- When the realization of a modelling method is expected to result in an **application software/tool**, a domain expert's (i.e., method developer) viewpoint need to be "augmented" with the viewpoint of a software developer (i.e., method engineer)

- Typically, a method developer **rarely considers** design, implementation or deployment relevant artefacts when "conceptualizing" a modelling method

- A method engineer on the other hand is usually **not an expert** in the domain that is addressed by a certain modelling method

# Praxis Sample: The BEN Instance

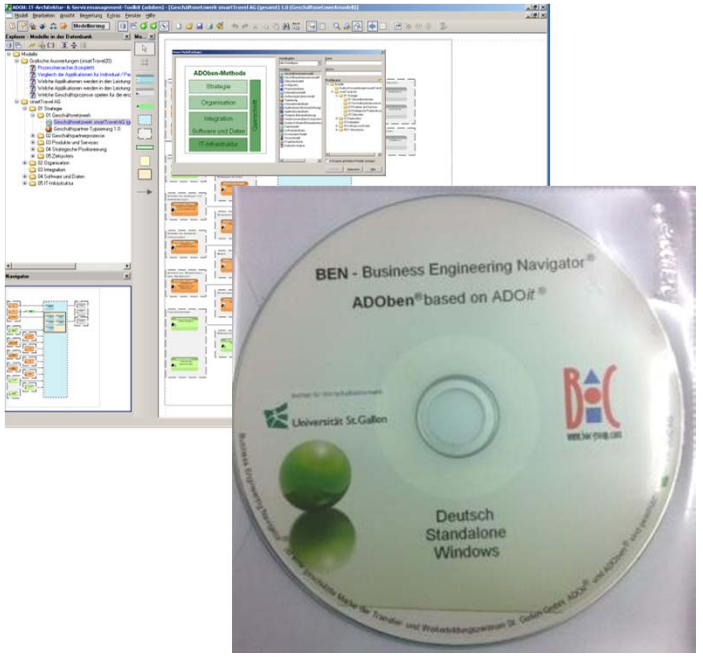
Step I → Step II → ..... → Step X




**Method Specification**

- Model Types
- Language Definition
- Functions
- User Behaviour
- …

?




Source: Winter R., "Business Engineering Navigator", Springer 2011

# Agenda: Conceptual Foundations

How do we like to do that?

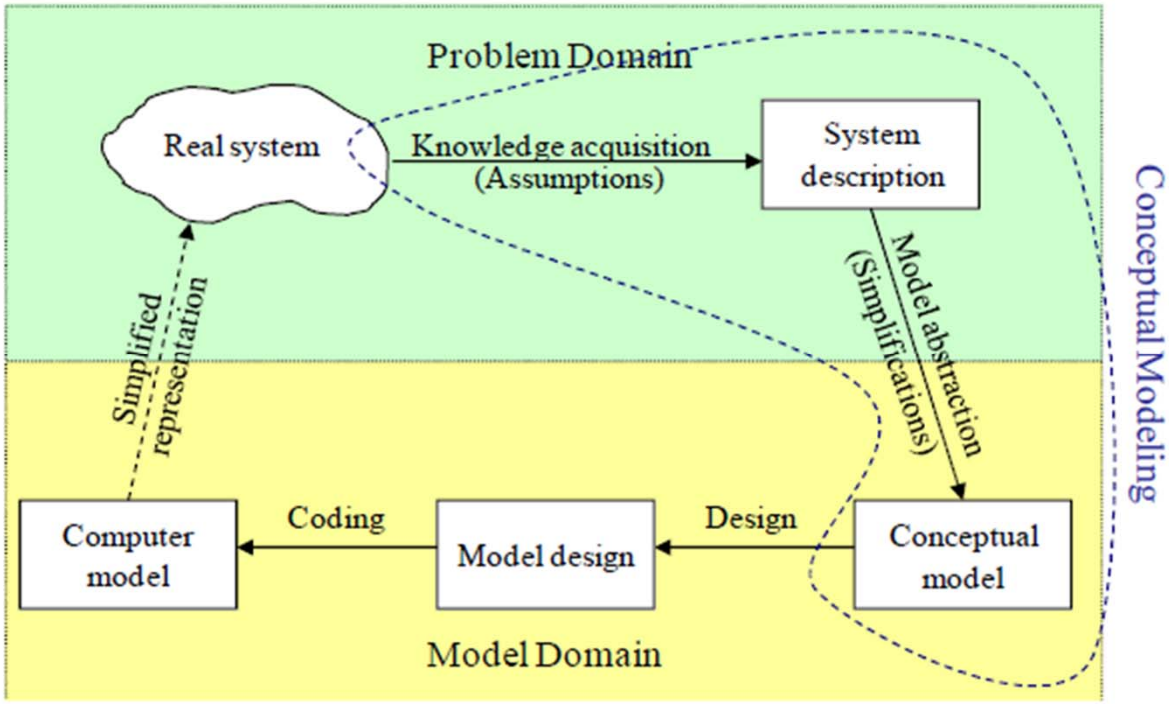
Proposed Approach: **„Meta-modelling" as a concept**

*A "Meta-modelling"* as an idea is introduced to rise the level of abstraction and to simplify the development of modelling languages, modelling methods, and finally, modelling tools.

# Why Metamodel !?

- Understand and describe the problem domain.
- Define a vocabulary for the elements in this domain.
- Help other understand the problem domain by using the same language.
- Manage complexity by raising the level of abstraction at which we think and design.
- *Additional functionality* for a specific domain of application should be engineered upon the meta-metamodel of the metamodelling platforms. That way a new generation of *more specialized* platforms will emerge.

# Conceptual Model




- *Conceptual model*, also known as domain model, represents concepts (entities) and relations between them, and is independent of design or implementation concerns.

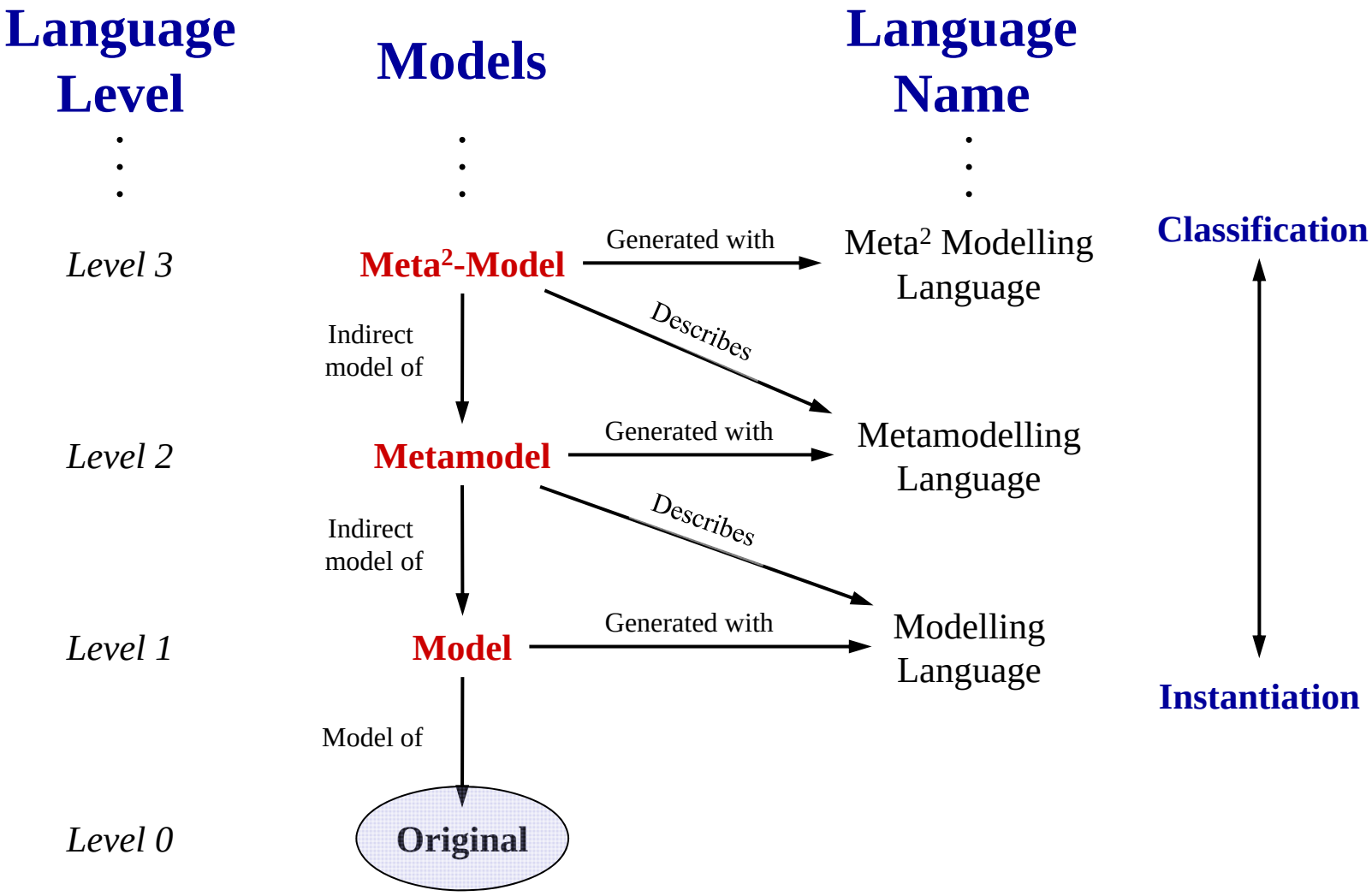
- Expresses the meaning of terms and concepts used by domain experts to discuss the problem, and to find the correct relationships between different concepts.

Robinson, S.: Designing Simulations that are better then the Rest: Conceptual Modelling for Simulation. In Proceedings: YoungOR 17, 5 – 7 April 2011

# A DSL for Modelling Methods

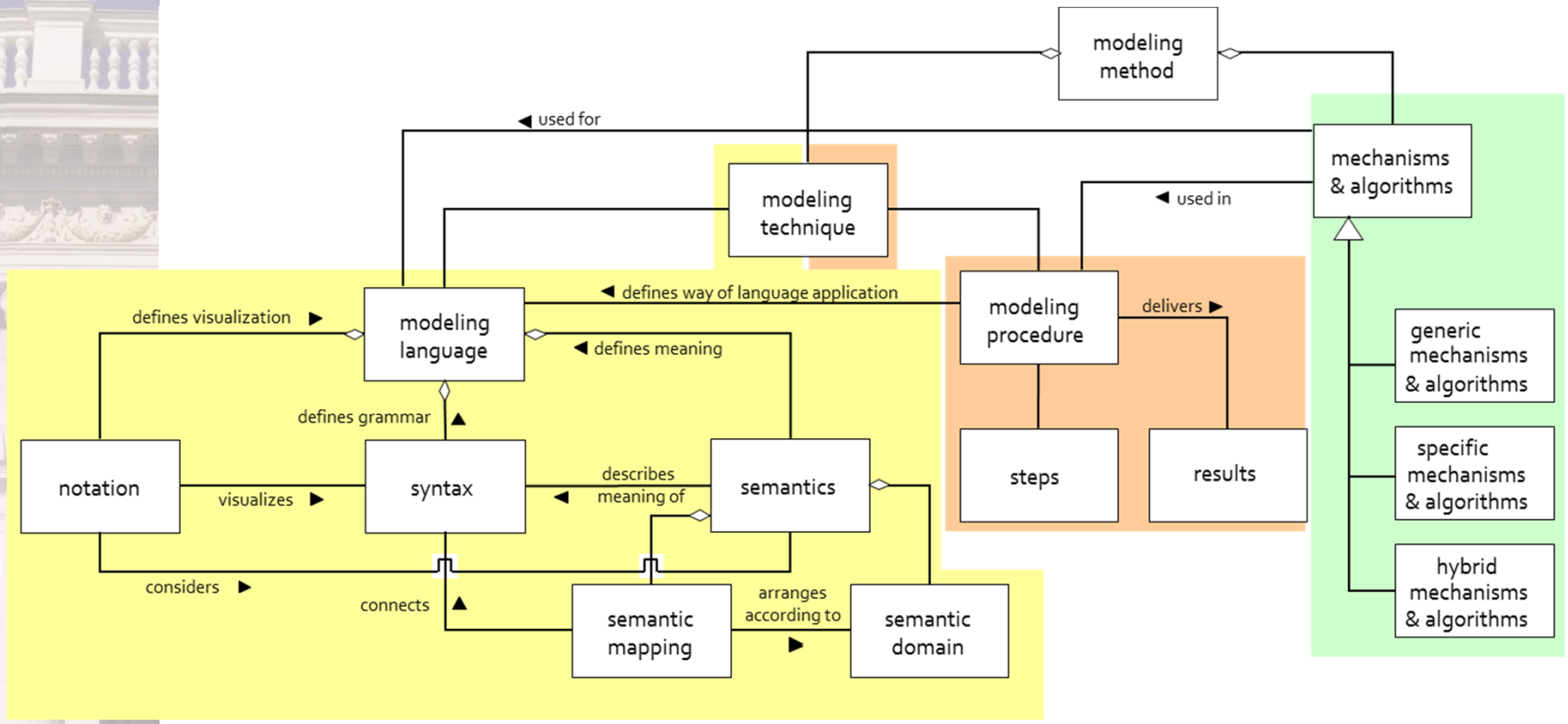
- Languages are the primary way in which system developers communicate, design and implement systems.
- Benefit of *metamodelling* is its ability to describe languages in a unified way:
  - Mappings can be constructed between any number of languages provided that they are described in the same metamodelling language.
- *Language engineering "DSL for ME"*
  - The effort that goes into producing a language definition can be overwhelming. By *reusing*, rather then reinventing, it is possible to reduce the time spent on development.

# Language Level: Model Hierarchy



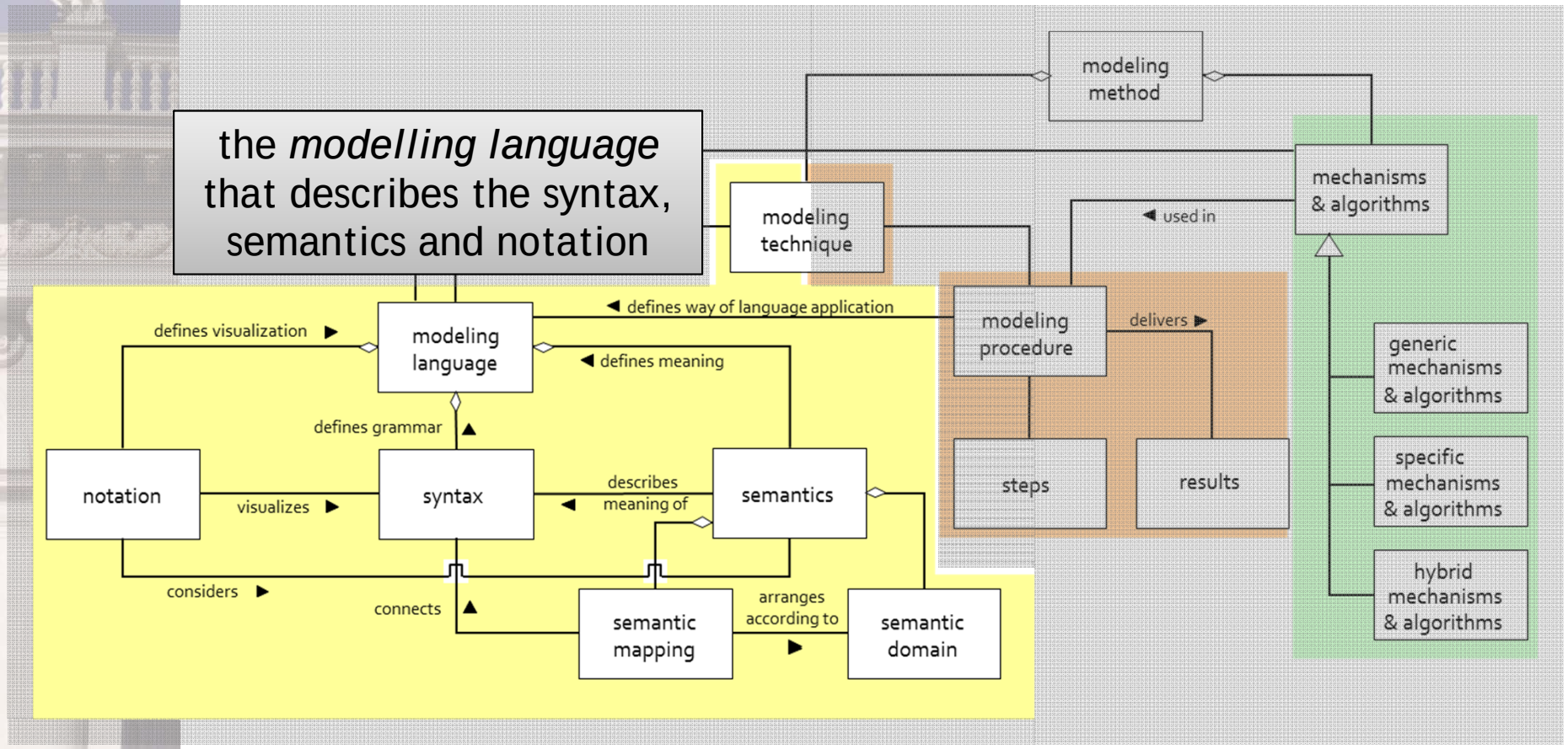
# Generic Modelling Method Specification Framework

Describes modelling methods on three major parts:



# Generic Modelling Method Specification Framework

Describes modelling methods on three major parts:



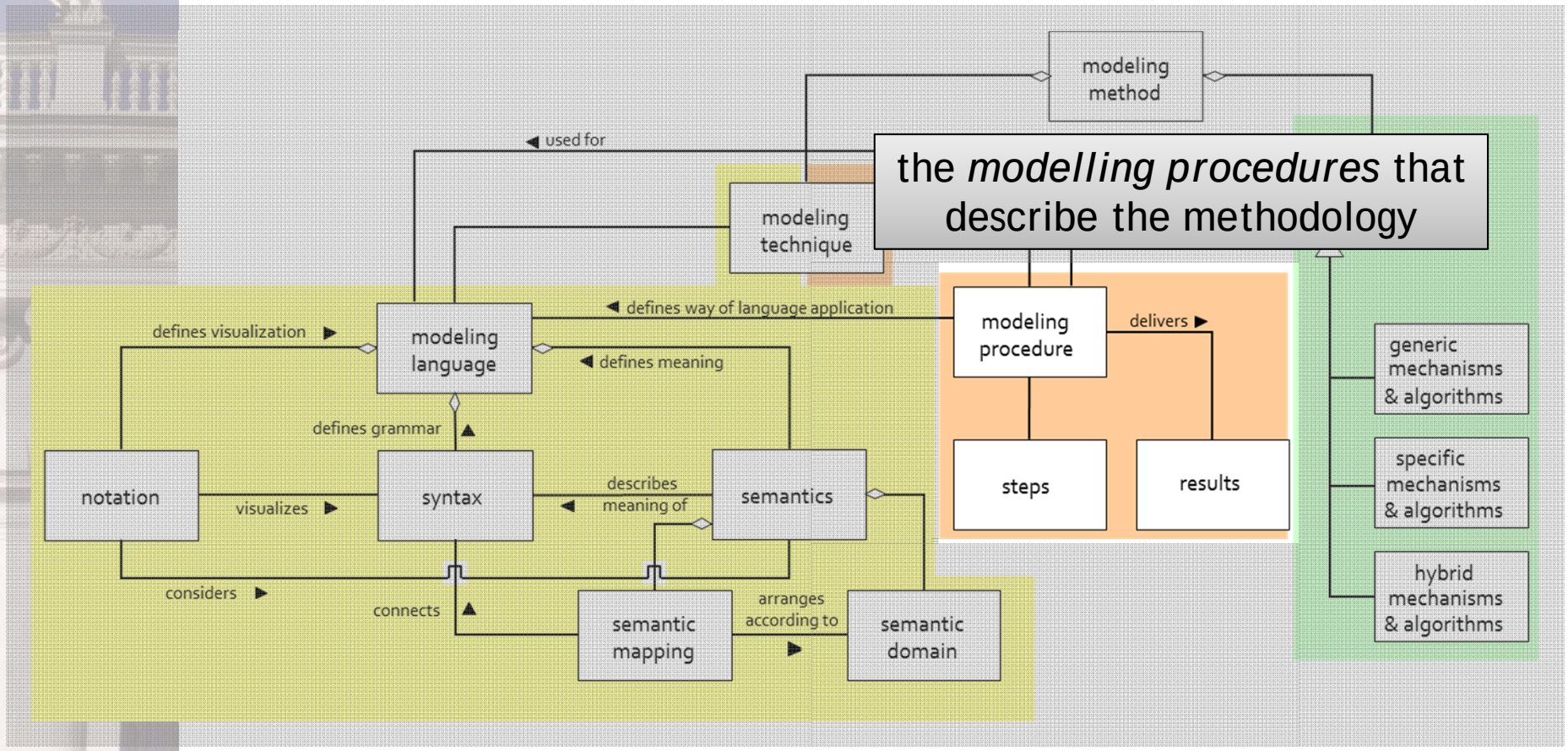
# Modelling Language:
## Semantics for Syntactic Elements

- Aspects of a modelling language that **cannot** be described with mechanisms for syntax definitions are pushed into the semantics area[1]
- Operational Semantics
  - The basic interest is on the “execution” of models based on an abstract machine
  - Eg., Interpreter for Petri-Nets or Statecharts
- Denotational Semantics
  - The denotation is expressed through a mapping of syntactic constructs to constructs of a commonly accepted domain that is assumed to be well understood
  - Eg., Control-Flow of BPEL denoted in terms of Petri-Nets

1) cf., David Schmidt, Denotational Semantics: A Methodology for Language Development, 1986

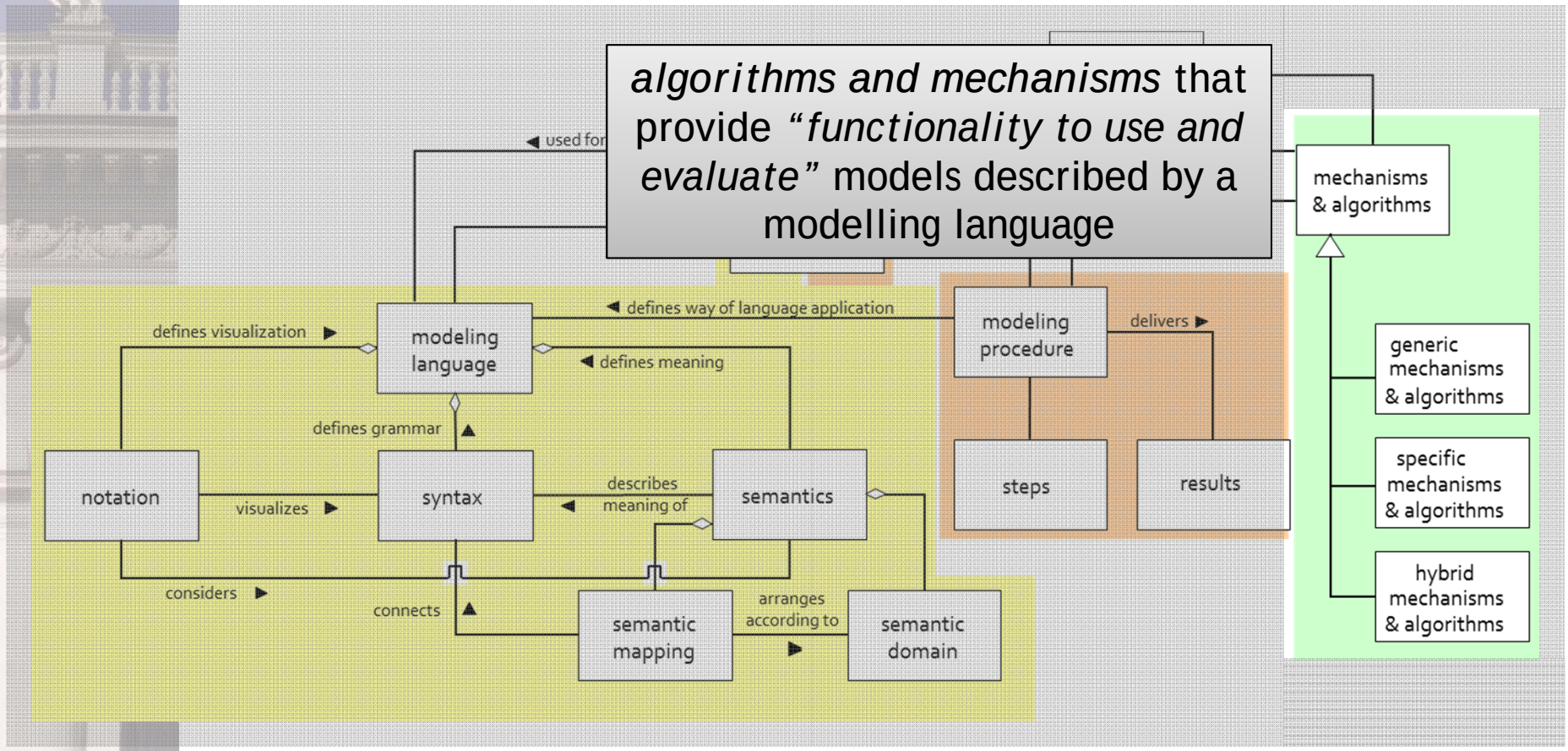
# Generic Modelling Method Specification Framework

Describes modelling methods on three major parts:



# Generic Modelling Method Specification Framework

Describes modelling methods on three major parts:



# Business Process Modeling Languages – An Overview

BP Design

## Integrated Definition Methods IDEF3

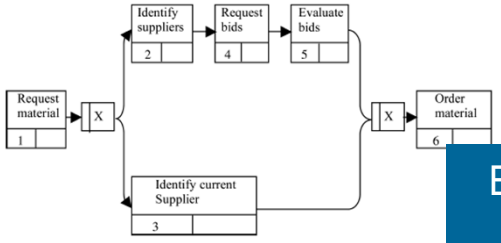



Fig. 5. A process diagram in IDEF3 (adapted from Mayer et al. [1995]).

## Petrinets

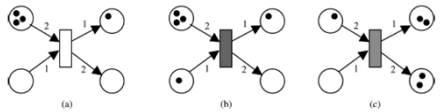
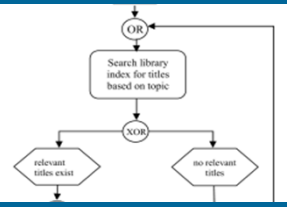
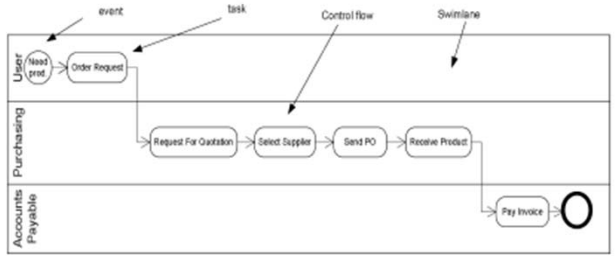



Fig. 7. A Petri net: (a) the transition is not triggered because the bottom left place does not have a token; (b) the transition is now triggered with a token moving in the bottom left place; (c) after firing, each input (output) place loses (gains) an equal number of tokens to the weight of the arc.

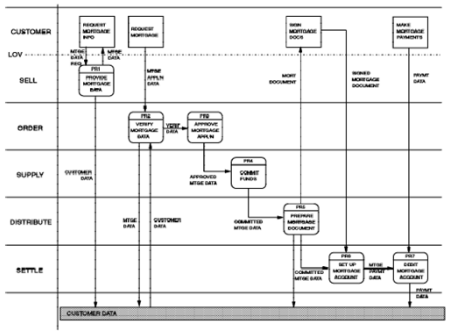
## Event Driven Process Chains (EPC)




## Business Process Modelling Notation (BPMN)




## LOVEM




## Role Acitivity Diagrams (RAD)

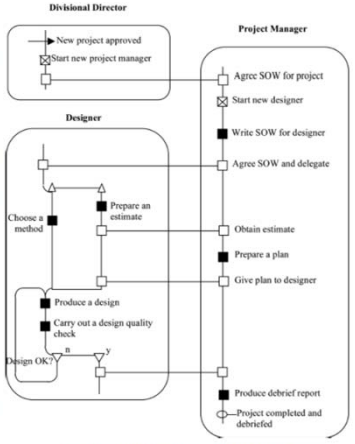
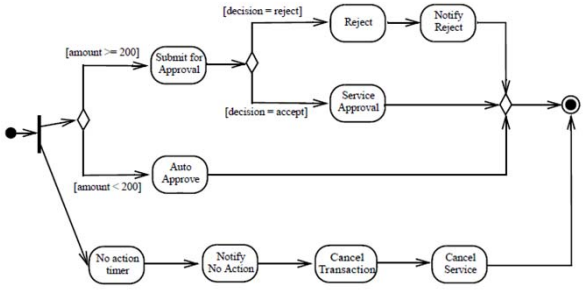
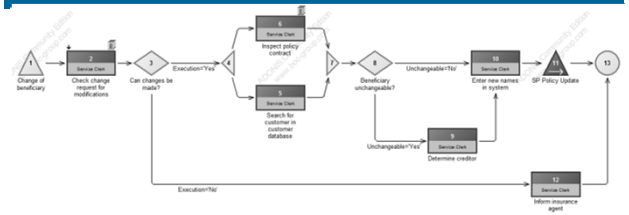



Fig. 8. Role activity diagram for carrying out a project.

## UML Activity Diagram



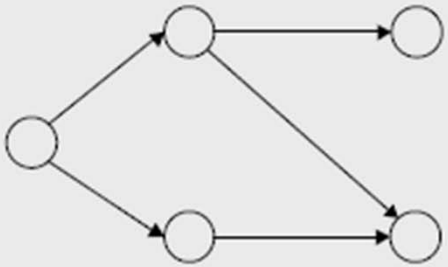
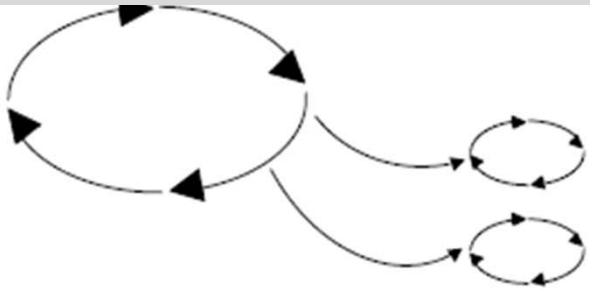
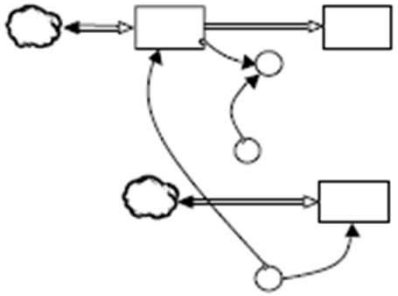

## ADONIS BPMS



**Source:** UML AD [OM10], LOVEM [IB95], ADONIS BPMS (sample models ADONIS), remaining [MT10]

# Business Process Modelling Languages - Types

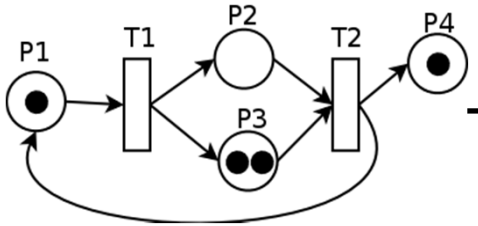


| Graphbased Languages | Rulebased Languages |
|---|---|
|  | IF (Activity of ?Current.Process is ?Send.product.to.customer) (Activity of ?Current.Process is ?Check.customer.creditability) . . . THEN (a.Subsequent.Activitiy of ?Send.product.to.customer is ?Check.customer.creditability) |
| **Speechact based Languages** | **Systemdynamic based Languages** |
|  |  |

**Source:** adapted after [Ju00]

# Apply a Modelling Method: Examples

**Petri Nets**

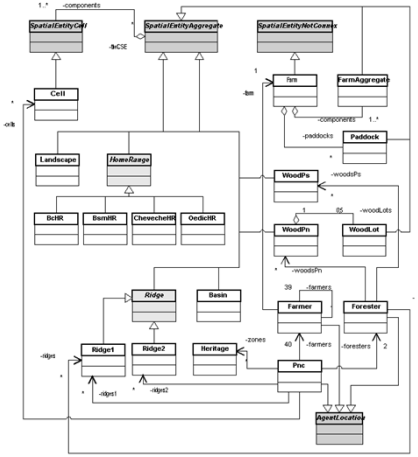



**Computer Hardware Engineer**

**Integrated Circuit**

Material

**UML**

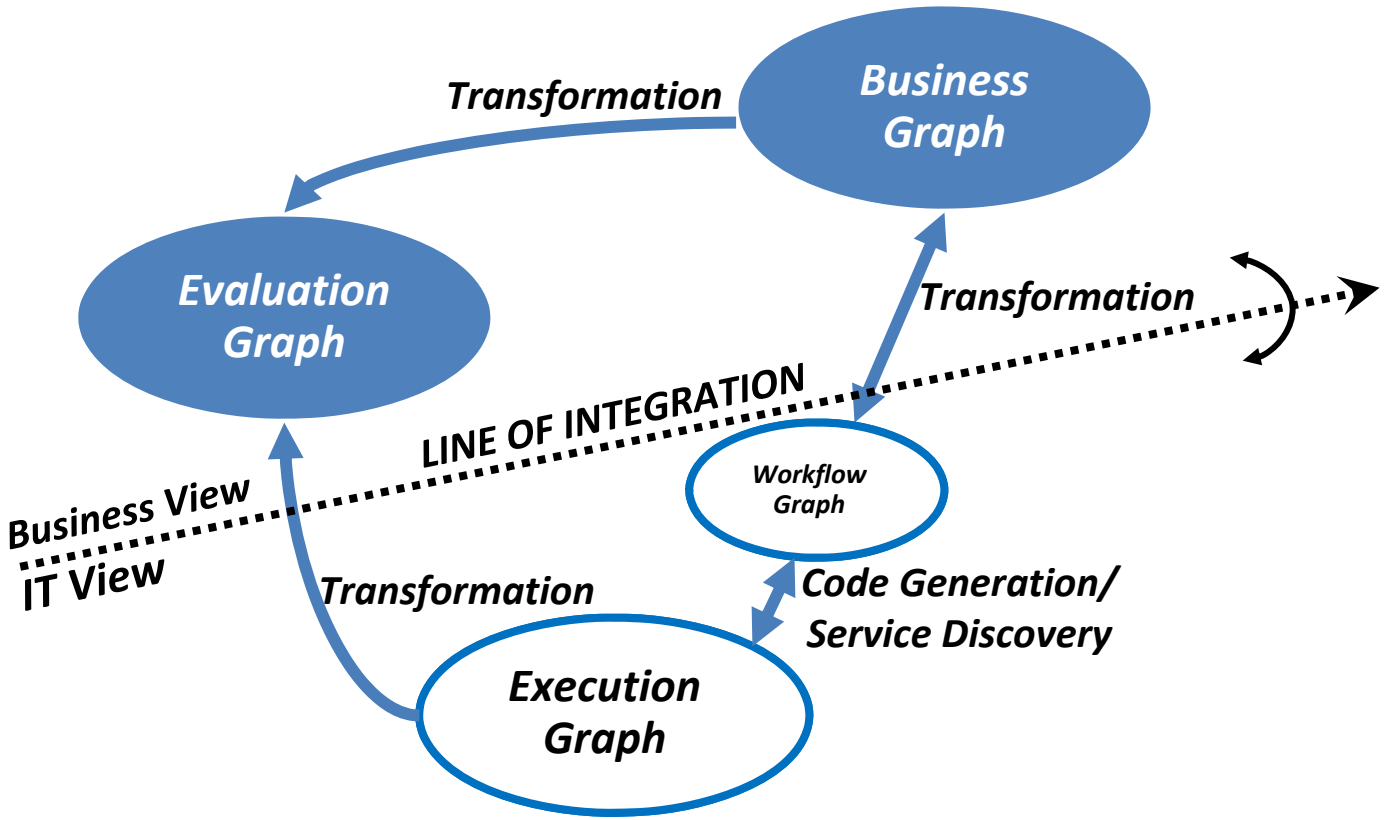


**Computer Software Engineer**

**Software**

Immaterial

# Apply different Modelling Methods

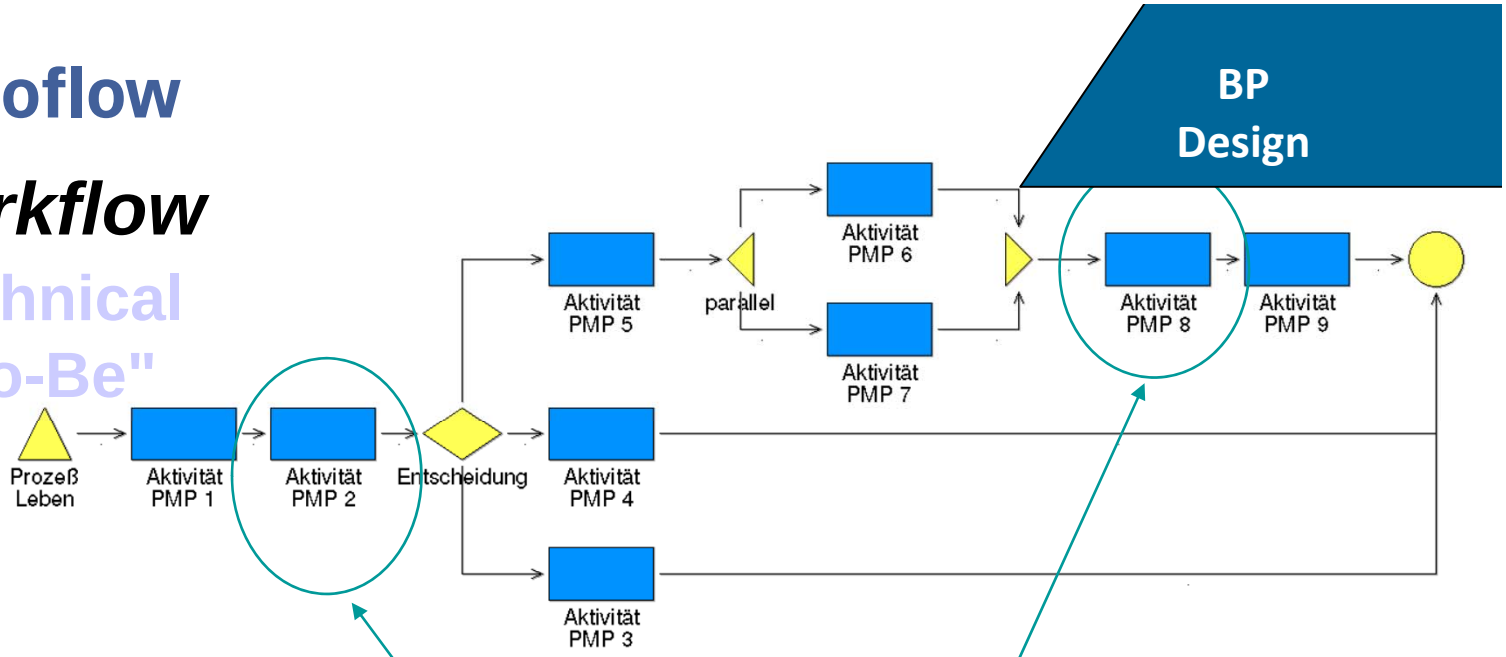



**Source:** adapted after [KJ96]

# Microflow

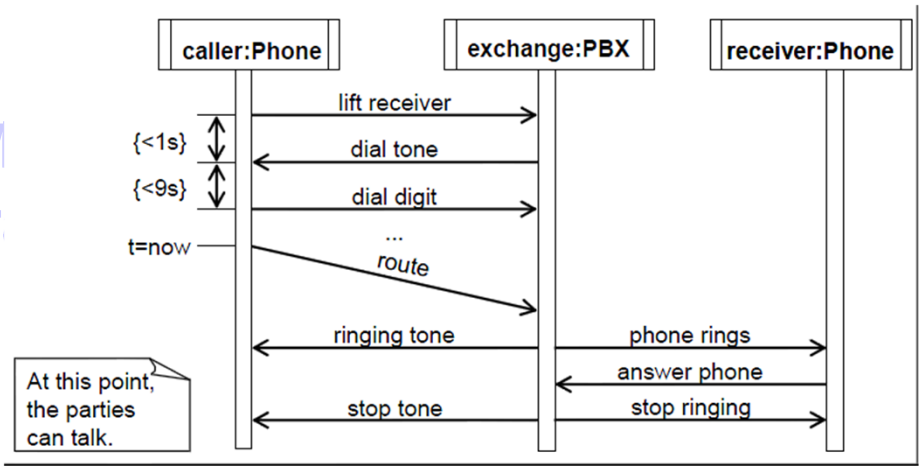
## *Workflow*

Technical
"To-Be"




## *Microflow*

Technical
Details, e.g. UM
Sequence Diagr



# Hybrid Modelling

- Fundamental *integration problem* among metamodels (modelling languages):
  - *Vertically different* (they vary in the level of details they describe);
  - *Horizontally different* (concepts on the same abstraction level describe different aspects);
  - Both vertically and horizontally different metamodels.
- There is a need to overcome *syntactical*, *structural* and *semantic* discrepancy of metamodels, in order to join their concepts together.

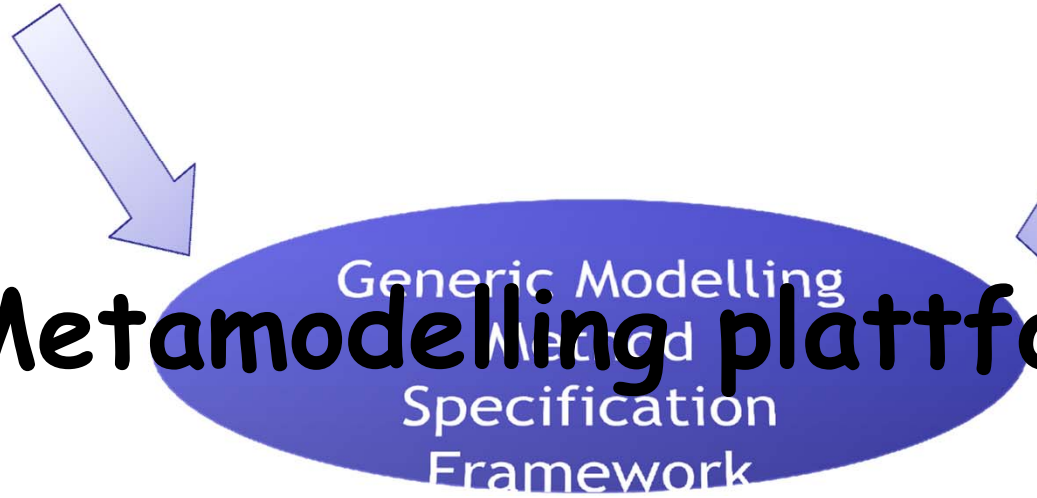
# Hybrid Modelling: Heterogeneity

- *Syntactical heterogeneity*
  - Represents the difference in formats intended for the serialization of metamodels.
- *Structural heterogeneity*
  - *Representational heterogeneity*: metamodels are represented using different metamodelling languages, each of them showing difference in its expressive power of available modelling primitives (classes, attributes, …);
  - *Schematic heterogeneity*: equal concepts are modelled either with different modelling primitives or with different number of primitives.
- *Semantic heterogeneity*
  - Difference in the meaning of the considered metamodel concepts.

# From Conceptual Foundations to Modelling Methods

Future enterprise systems require an elaborate conceptual foundation that promotes a tight mutual alignment between information systems and business to effectively support business operations and managerial decision-making.

Thus a growing number of groups around the world show interest in modelling methods - either standard or individual ones - that satisfy the requirements of their domain and comply with the conceptual foundations.




**Metamodelling plattforms**

# Agenda: The Approach

Which approach do we like to use?

Metamodelling plattforms*

*Karagiannis, D., Kühn, H.: „Metamodelling Platforms“.
In Bauknecht, K., Min Tjoa, A., Quirchmayer, G. (Eds.):
Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer, Berlin/Heidelberg, p. 182 ff.

# Metamodelling Platforms: Some Features

- Extensible, repository-based metamodelling platform
- Three-step modelling hierarchy with a rich meta-metamodel
- Can be customized using metamodelling techniques
- Extendable with custom specific components
- Platform kernel provides basic modules for managing models and metamodels
- Graphical and tabular model editing
- Scripting language for defining mechanisms and algorithms

# Hybrid Modelling: Platform Support

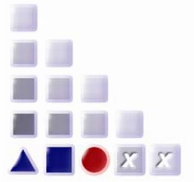
- Metamodelling platforms should be realized on a component-based, distributable, and scalable architecture.
- The meta-metamodel, most important element of the platform, needs to define all the necessary concepts.
- The model repository needs to be designed to accommodate the reuse of already developed modelling method constructs.
- Hybrid modelling methods can be developed using chunks and pieces from the repository by binding them together using appropriate mapping and integration rules.

# Metamodelling Environments: An Overview

In general, metamodelling environments can also be used to specify and implement “domain-specific” modelling tools.

Metamodelling Platforms:
- ADO*xx*
- MetaEdit+
- Obeo Designer
- GME
- ConceptBase
- …

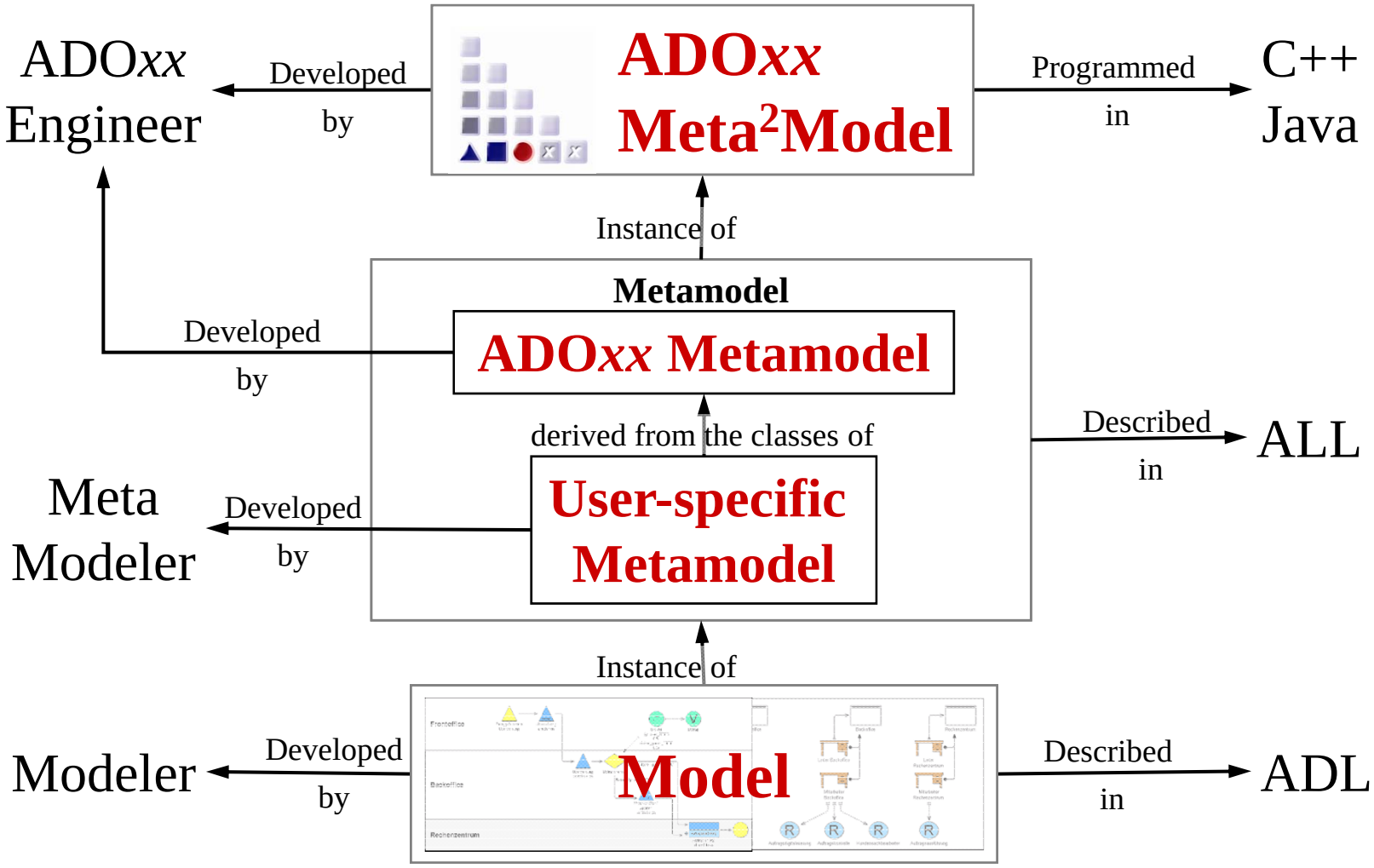
Metamodelling Frameworks:
- Eclipse: EMF (GEF, GMF), and others…
- Visual Studio: Visualization & Modeling SDK
- …




# What is ADO*xx*?

**ADO*xx* is a metamodeling development and configuration platform for implementing modelling methods.**

# Metamodelling Platforms Hierarchy: ADO*xx*®

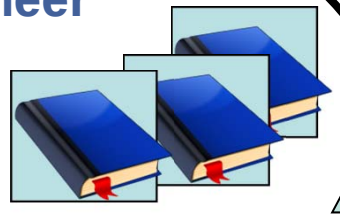


# Approaches

"Conceptual Modelling"

Method Engineer

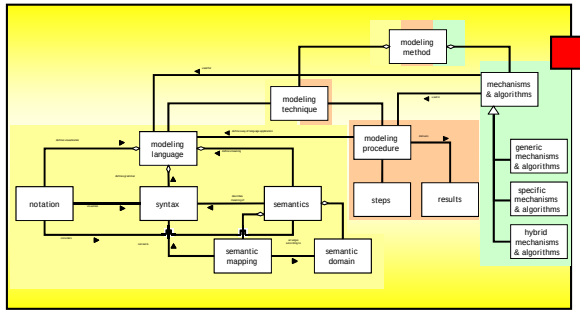
Top Down



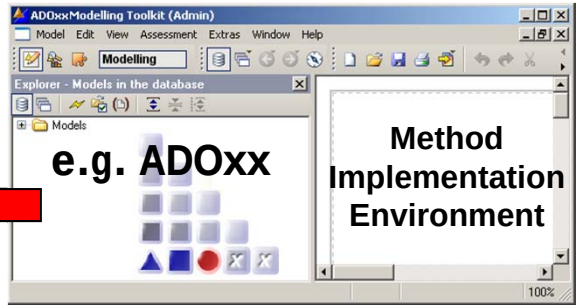
Input: generic

Theoretical Method Development

*Conceptualization*



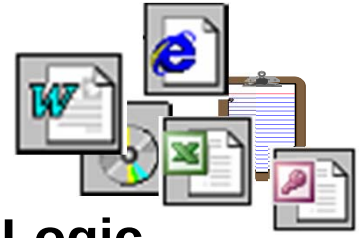

**Conzeptualization of the Method**




*Validation*

Bottom Up

Input: specific, like Data, Information



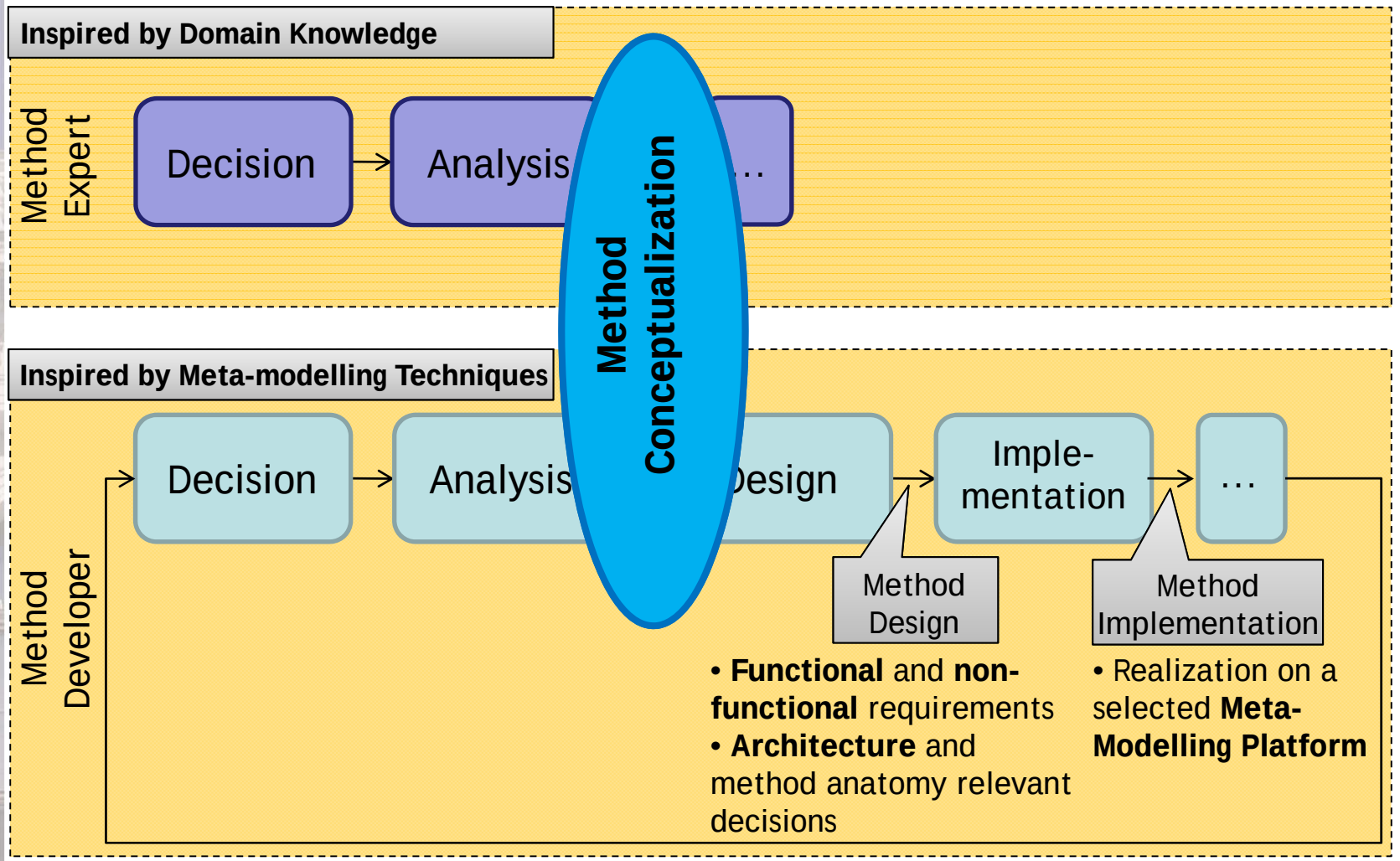
Business-related Abstraction of a Development Environment

**Domain: Business Logic**

# Graphical Representation of implemented Metamodels

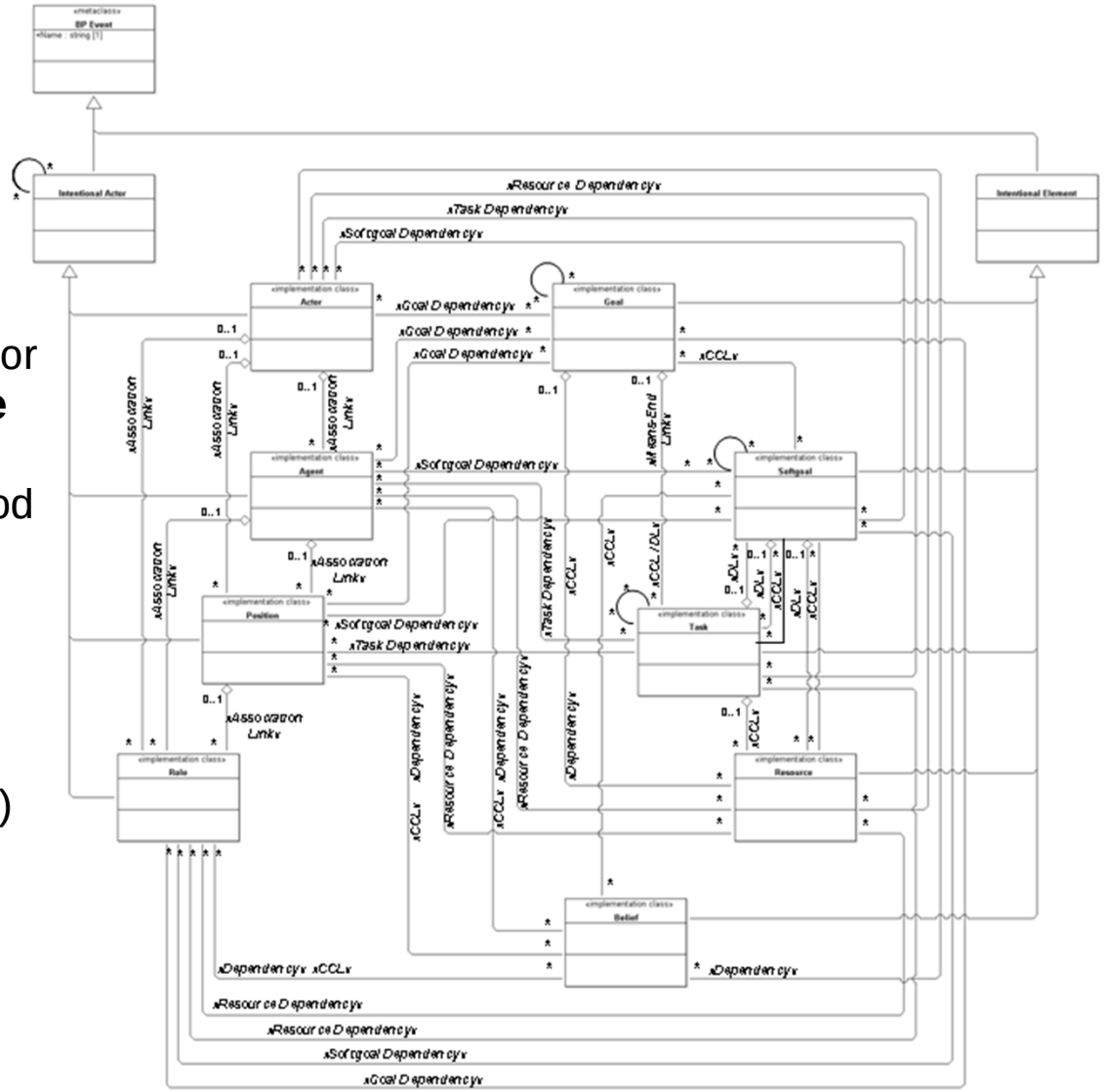
- Why?
  - Human-oriented representation of implemented modelling method concepts
    - > Understandability
    - > Evaluation
    - > Refactoring
- How?
  - Selection of an appropriate language or formalism **to describe / represent** a Metamodel
  - Description of method concepts in terms of elements from the selected language or formalisms (i.e., a mapping to their notation)

# Realizing a Modelling Method
## (Simplified view)

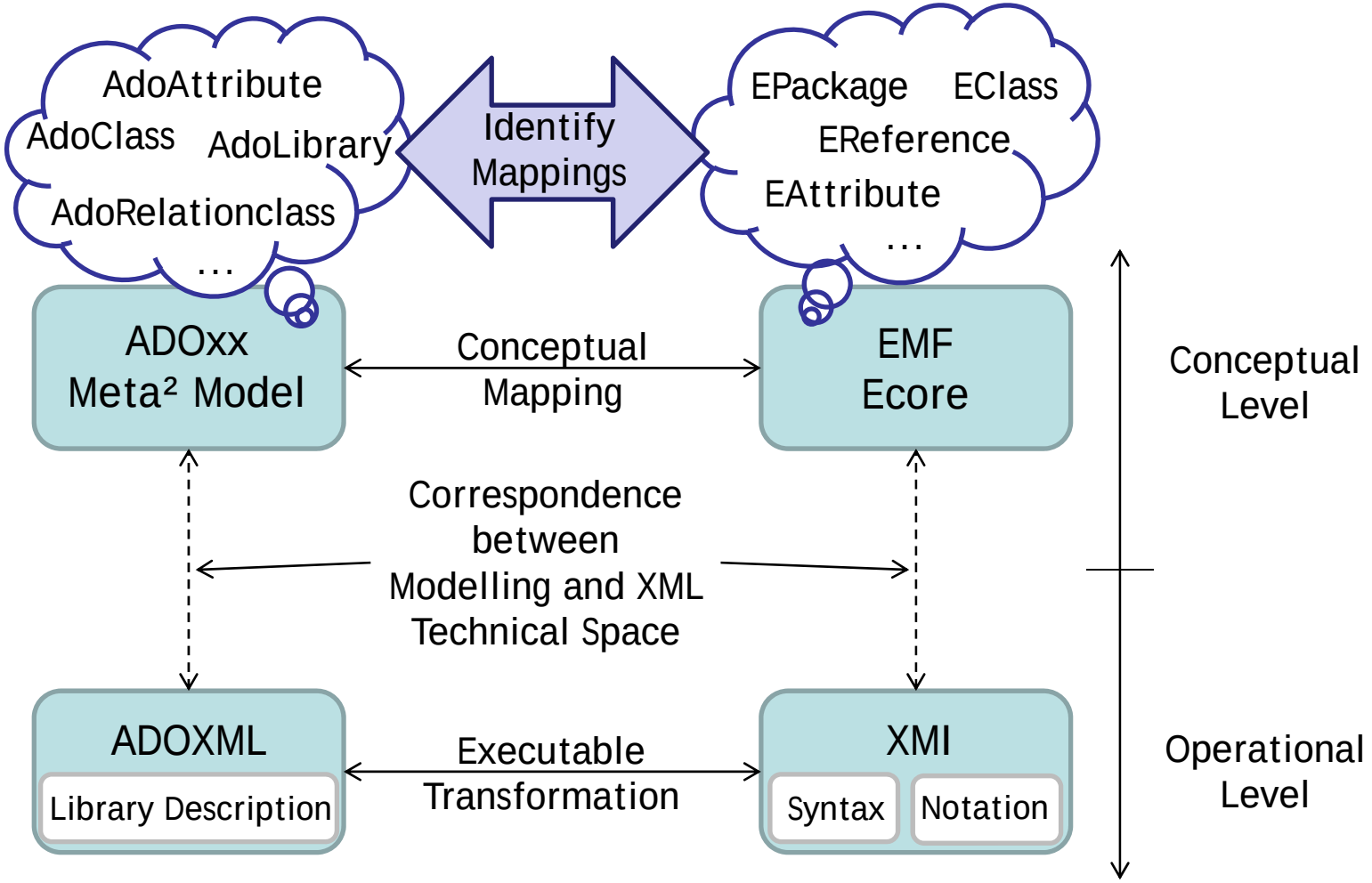


# Functionality Example: The Metamodel of i*Star

- Selection of an appropriate language or formalism **to describe** the Metamodel
- Description of method concepts in terms of elements from the selected language or formalisms (i.e., a mapping to their notation and meaning)



# Experiment – Representation of ADOxx Library Elements in UML Notation (1/2)



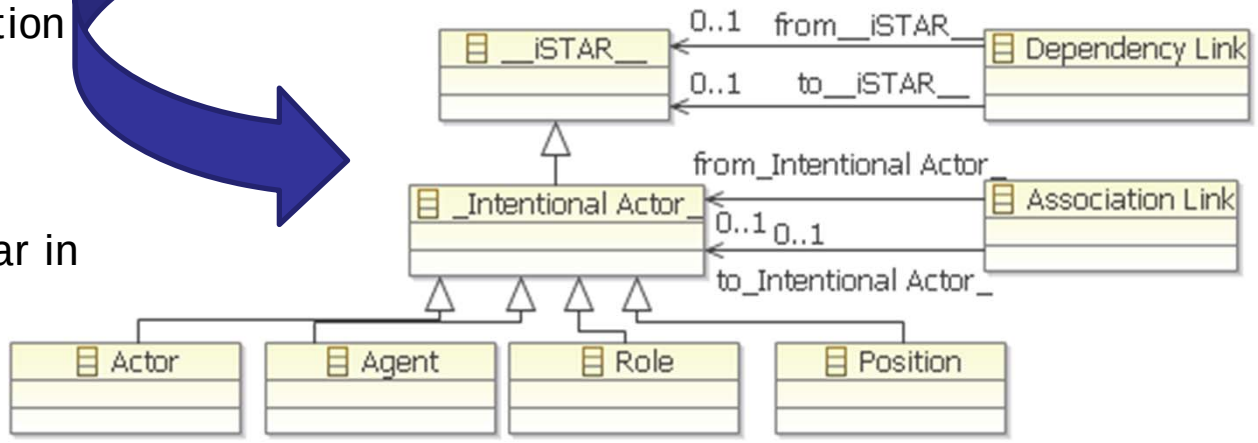
# Experiment – Representation of ADOxx Library Elements in UML Notation (2/2)

```
<library language="en" name="iSTAR Method_v1.00_for ADOxx v1.0_20091016"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AdoXmlSchema.xsd">
  <classes>
    <class name="_Intentional Actor_" superclass="__iSTAR__" libtype="bp">
    <class name="Actor" superclass="_Intentional Actor_" libtype="bp">
    <class name="Agent" superclass="_Intentional Actor_" libtype="bp">
    <class name="Role" superclass="_Intentional Actor_" libtype="bp">
    <class name="Position" superclass="_Intentional Actor_" libtype="bp">
    <class name="_Intentional Element_" superclass="__iSTAR__" libtype="bp">
    <class name="Goal" superclass="_Intentional Element_" libtype="bp">
    <class name="Softgoal" superclass="_Intentional Element_" libtype="bp">
    <class name="Task" superclass="_Intentional Element_" libtype="bp">
    <class name="Resource" superclass="_Intentional Element_" libtype="bp">
```

Extract from iStar ADOxx Library Description

Transformation

Extract from iStar in terms of an UML Class Diagram



# Classification of Metamodelling Platforms

- Generally, we can divide metamodelling platforms in two groups:
  - The ones that specialize in *model-based code generation* (MCG);
  - The ones that specialize in *model analysis & simulation* (MAS).

- Other comparison can be based on value added to the platform:
  - Extra features that are distinguishing one platform from the other.

# Model-Based Code Generation

- MCG (model-based code generation) metamodeling platforms:
  - Support MDE (model-driven engineering) methodology;
  - Automated transformations of source models into complete code, that can be compiled and interpreted for execution;
  - Generally restricted to developing only certain kind of applications;
  - Restrictions come from high domain specialization (not only modelling language, but also underlying framework and code generator);
  - Focusing on a narrow area of interest makes full code generation realistic (very difficult to achieve with general-purpose modelling languages – UML, etc.).

# Model Analysis & Simulation

- MAS (model analysis & simulation) metamodelling platforms:
  - Support EM (enterprise modelling) methodology;
  - Models primarily used for analysis and simulation of business processes, to find the means to improve their efficiency and quality;
  - Additionally, models are used for sharing of knowledge;
  - Platforms are specialized for creating modelling methods (upgrade on modelling languages, including procedures, algorithms & mechanisms).

# Agenda: The Applications

# Enterprise Modelling: The ComVantage Project

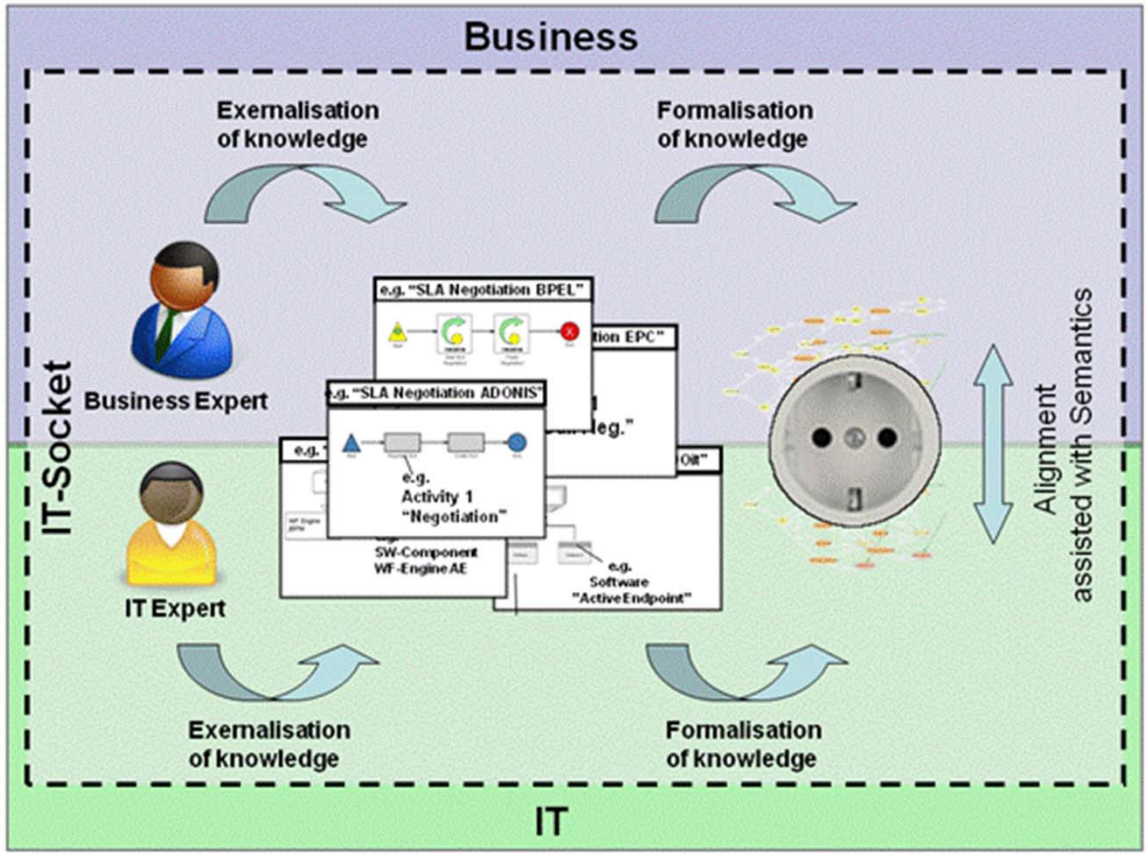

**ComVantage** will identify:

- what bits of information are relevant for sharing collaboratively and between which parties;
- reducing the extracts to amounts that are not valuable for misuse.

- Aims at providing a product centric information space for cross-organizational information that is shared during production time and beyond.

http://www.comvantage.eu

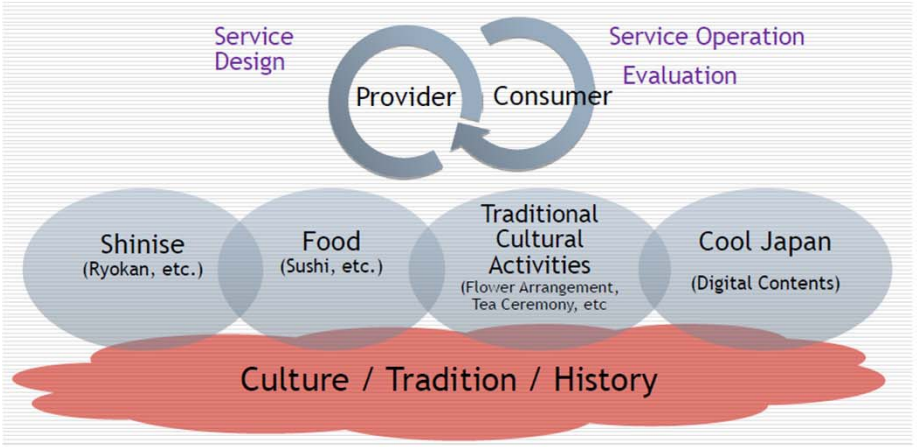
# Service Modelling: The PlugIT Project




Knowledge within the experts head is externalized in models and further formalized to enable automated support of business and IT alignment, which can be delegated to semantic technology.

http://plug-it.org

# Cultural Modelling: The JCS Project



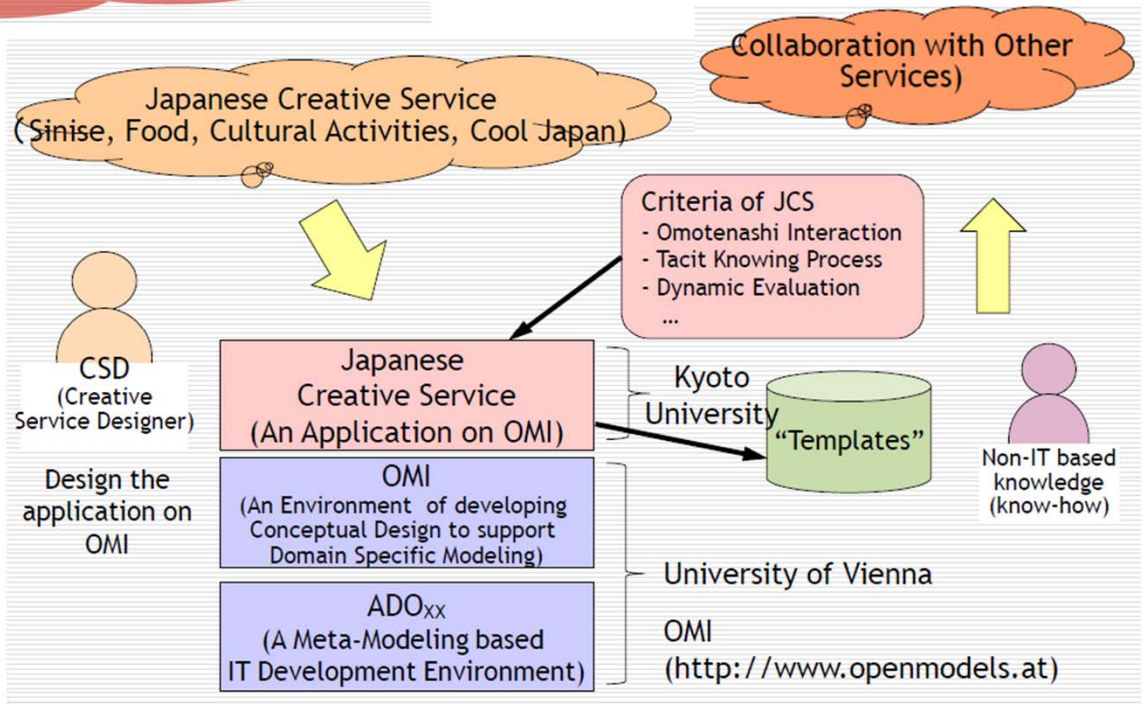

Develop the fundamentals of sustaining and developing value on service based on culture

Adapted from: Hara, Yoshinori, "Theoretical Analysis of Creative Service Management and its Application to Open Model Initiative", Project Presentation Slides, September 2011

Flexible Enhancement based on metamodelling.

Optimization (balancing) of the hybrid approach.

Seamless integration with IT development environment.



# Agenda: The Evaluation

**A scientific:** The Open models initiative

www.wikimethods.org

**A business:** The BOC-Management Office

www.adonis-community.com

A spin-off from the University of Vienna

# The Open Models Initiative

- The initiative based-on an *open-membership for all interested researchers and academic organizations*.

- … an *international scientific community* working on establishing this initiative, which focuses on the creation, design, evolution and processing of modeling methods and the models designed within them.

# The Open Model Initiative Works Through …

## Community

Groups of individuals sharing common values and following common goals.

Organized in communities of practice for different domains providing value through i.e. competence, joint activities, shared practices and resources, sustained interaction, experiences, and tools.

## Projects

Modeling Environment Projects: creating model content for various domains and/or purposes.

Method Engineering Projects: conceptualization of new or further development of existing methods, development and deployment of IT-based modeling tools.

## Foundations

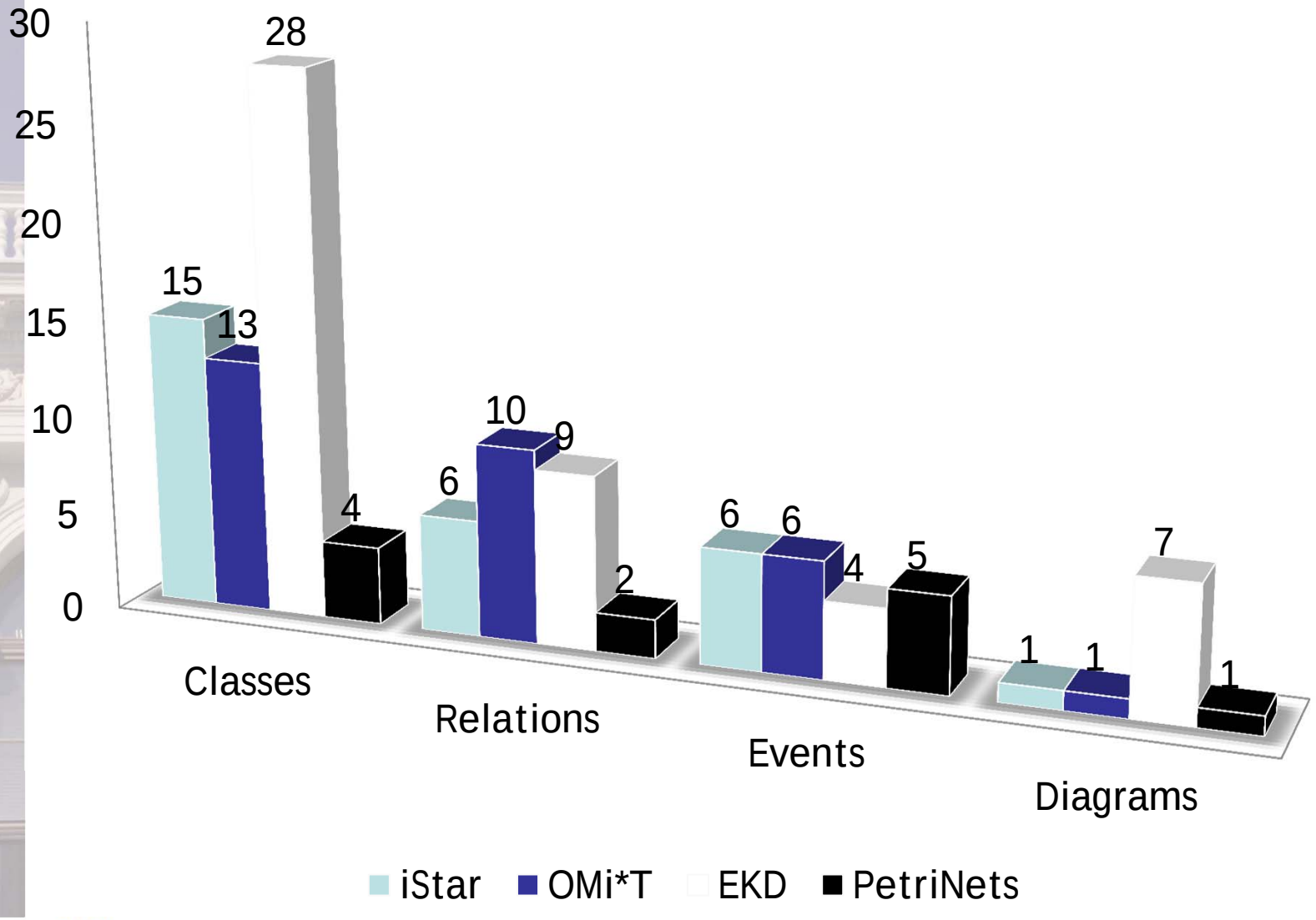
Modeling languages and their algorithms for the processing of models as well as IT-based modeling environments.

Enabler that supports designers to choose the right algorithms for the processing of methods and models.

# OMI-Portal: Selected Modelling Methods

| Modelling Method / Language | Application Domain |
|---|---|
| **EKD** | Enterprise modelling (requirements for change) |
| UML | Software design |
| **iStar** | Requirements engineering (agent oriented approach) |
| **BEN** | Holistic design and management of organizations |
| **PetriNets** | Information processing systems |
| KPNs | Signal processing |
| BPMN | Process modelling |
| **eGPM** | Exemplary business process modelling |
| SysML | System engineering |
| **MeLCa** | Large scale collaborative processes design |
| **SOM** | Business systems (capturing business semantics) |

# ADOxx®: Method Building Blocks



# OMI: How to participate

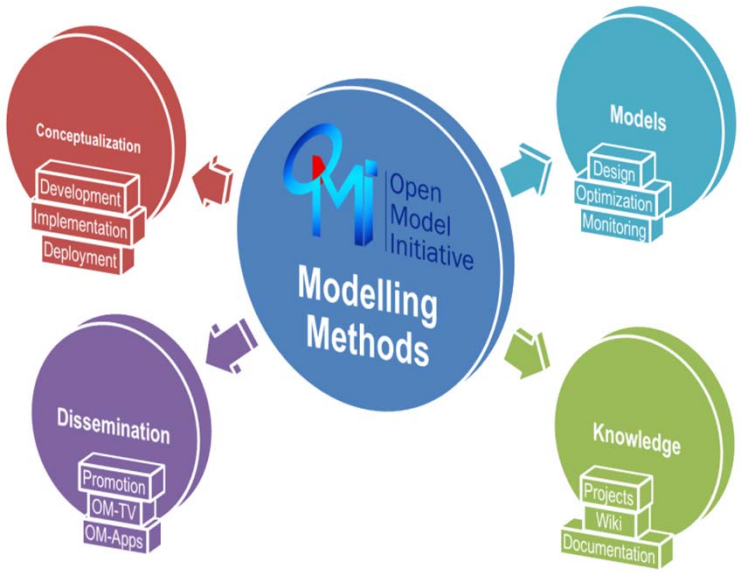
## Projects

can be realised, either on:

- Community Level
  - Methods
  - Models
- Individual Level
  - PhD-Thesis
  - Master Thesis

within the Open Models Universe.

Register at:
www.openmodels.at




# Some Research Issues and Conclusion

# Some Research Issues (I)

- *Alignment of business process and security* (prevention strategies against social engineering attacks, addressing security risks in business process modelling, security threats identification, etc.).

- *Optimizing information flow and efficient reuse of existing knowledge* as part of the business strategy of viable enterprises (approaches and solutions for active, viable, and agile information systems, information logistics and knowledge supply, etc.).

- *Intelligent educational systems* (collaborative learning environments, virtual and distant education, internet based tutoring systems, etc.).

# Some Research Issues (II)

- *Information integration* (event based data integration, user centric data integration, streaming data integration; solving information overflow problem for the users, etc.).
- *Interoperability* (completely understandable interfaces to share data between different systems, people, and businesses, etc.).
- New *architectures for information systems* (enterprise architecture frameworks, ERP development approaches, etc.).
- New *modelling methods*, *modelling* and *metamodelling tools*.

## Conclusion

*There are no bad modelling methods, but only not appropriate ones!*

*For Enterprise Information Systems one modelling method is not sufficient!*

*Hybrid modelling methods are required.*

# Thank You For Your Attention!


**Prof. Dr. Dimitris Karagiannis**
*University of Vienna*
Faculty of Computer Science
Research Group
Knowledge Engineering
Bruennerstr. 72
1210 Vienna
P: +43-1-4277-39580
F: +43-1-4277-39584
dk@dke.univie.ac.at

universität wien

**Dipl.-Ing. Niksa Visic**
*University of Vienna*
Faculty of Computer Science
Research Group
Knowledge Engineering
Brünnerstr. 72
1210 Wien
P: +43-1-4277-39589
F: +43-1-4277-39584
nv@dke.univie.ac.at