Model-based software and data integration for business applications

Dr. Ralf-Detlef Kutsche
TU Berlin and Fraunhofer ISST/FIRST/FOKUS

IT4BI/eBISS Summer School 2013
Dagstuhl, Germany, July 10, 2013

Bundesministerium für Bildung und Forschung

WACHSTUMSKERNE UNTERNEHMEN REGION
Die BMBF-Innovationsinitiative Neue Länder

Fraunhofer FOKUS

AGENDA of the eBISS Tutorial on MBSDI

- History & Background
- Goal: Model Based Software & Data Integration
- Languages and Models
- Metalanguages and Metamodels
- BIZYCLE: Model Based Software & Data Integration
- BIZWARE: Domain Specific Languages

My Modeling Background / History

- Lectures in Information & Software Modeling / Student Projects
  - Grundlagen der Informationsmodellierung (TU Berlin, since 1994)
  - Information Modeling (Univ Jönköping, 2003, 2004)
  - Advanced Modeling (TUB, since 2008; HCMUS, VietNam, since 2010)
  - Heterogeneous Distributed Information Systems HDIS (TUB, since 1996)
- Projects for Industry and Public Services (HDIS, since 1989)
  - Health Care (German Heart Center Berlin, TMF Clinical Studies, ...)
  - Environmental Information Systems („Landesumweltinformationssysteme“)
  - European Migration Network (Society: Migration & Asylum, Politics, ...)
  - Consultancy for Software Enterprises (Software Modeling, ...)
- Industrial Cooperation Projects: BIZYCLE & BIZWARE
  - Partners in the areas of: Health Care, Production/Logistics, Facility Management, Publishing, Finance, ...

Large Scale Information Infrastructures Specifications & Modeling ... my personal history ...

- PAtientenDatenKOMmunikation PADKOM (*Patient Data Communication*) / Heterogeneous Distributed information Management System HDMS at German Heart Centre Berlin DHZB (1989 – 1993+)
- LandesUmweltInformationsSystem LUIS Brandenburg (1994 – 2000+) (*State Environmental Information System Brandenburg*)
- Telematikplattform für die Medizinische Forschung in Deutschland TMF (1998 – 2003+) (*Telematics Platform for Clinical Studies*)
- European Migration Network EMN (2004 – 2006+)

PADKOM / HDMS @ DHZB

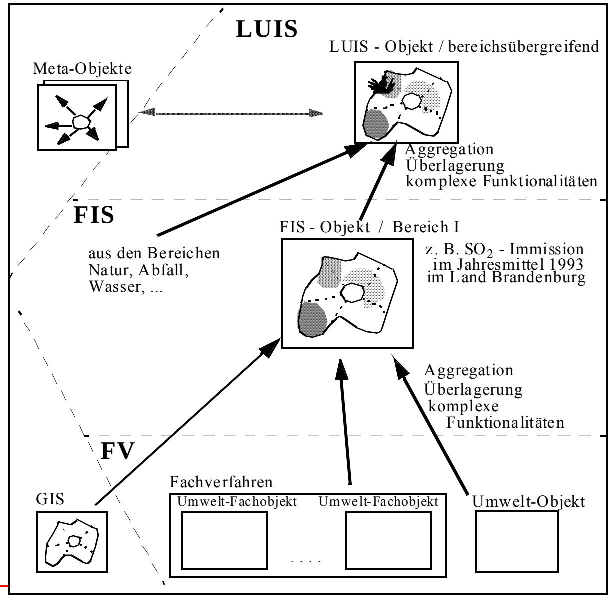
since 1989

- Development of the HDMS …
- … as an object-based, distributed, heterogeneous information system in the domain of cardiology at German Heart Centre of Berlin DHZB …
- … basically enabling communication and an integrated view of the various kinds of patient data … (later also: data integration)

LUIS Brandenburg

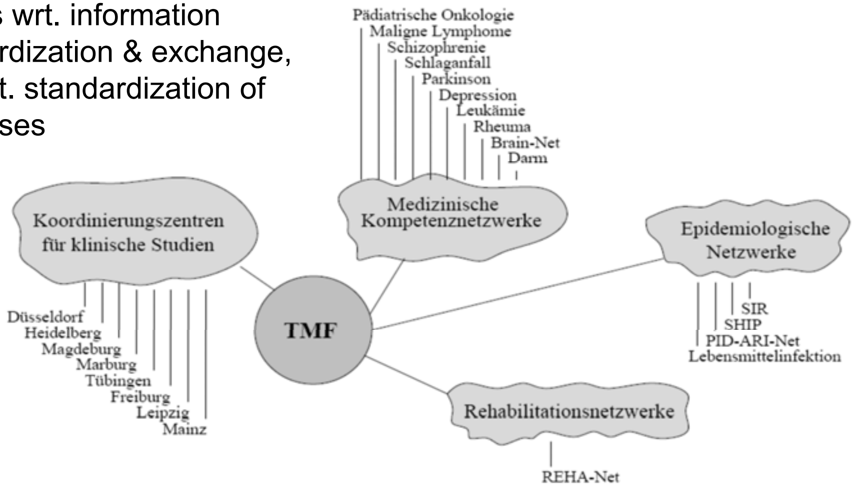
since 1994

- Integration of environmental information sources via object-oriented middleware …
- Object-based approach …
- Metadata-based approach …



TMF Germany since 1998

- Integrating approx. 200 medical centers performing clinical studies wrt. information standardization & exchange, and wrt. standardization of processes

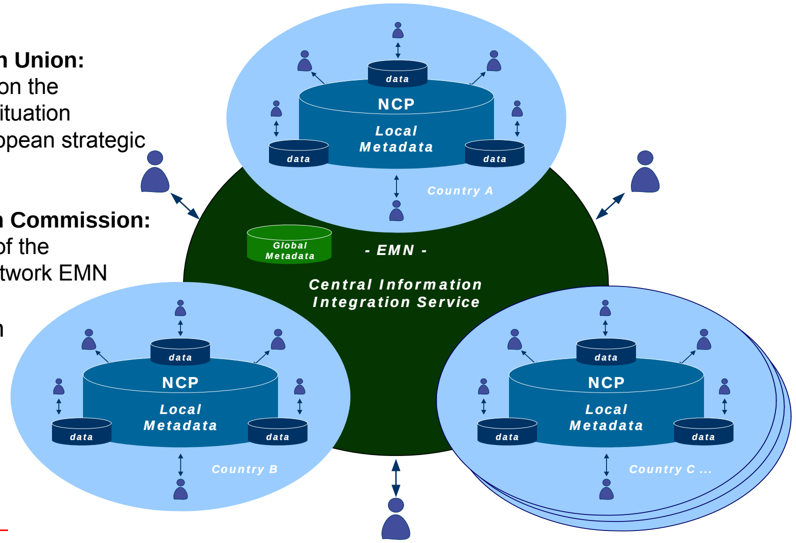


European Migration Network

since 2004

Goals of the European Union:
- Comprehensive view on the migration and asylum situation
- Harmonization of European strategic policy & legislation

Task for the European Commission:
"Pilot Implementation" of the European Migration Network EMN w.r.t.
- Contact Information
- Publications
- Legislation
- Statistics
- …



Lessons Learned from the Experiences …

- **Model-based development** is essential in order to guarantee flexible, stable and sustainable software solutions and information systems
- **Metadata definition, standardization and management** is essential in order to allow for data exchange and integration
- **Semantic concepts** help to improve ‚real' semantic integration
- **Graphical languages help** in faster perception of complex structures and relations and often help to avoid inception phase errors
- **Object-oriented paradigm** helps to unify structural (data) view and dynamic (functions, processes, interactions, workflows) views
- **Rich middleware platforms** are essential in order to allow for complex interoperability (e.g. transactions, security, …)
- **Solid mathematical / theoretical background** is absolutely useful

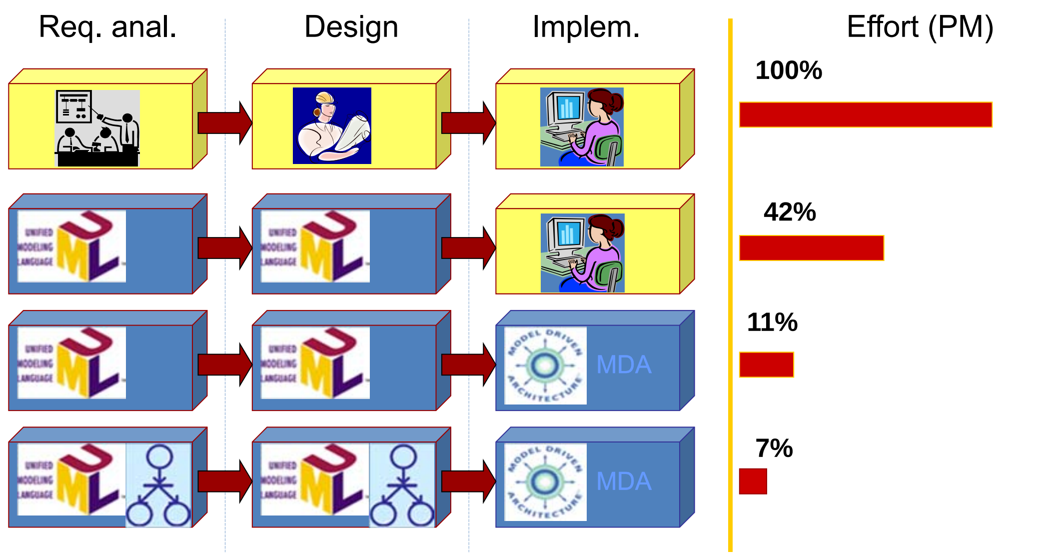
Motivation 1970 – 2070 … (?)

- Integration of heterogeneous distributed IT-systems is one of the major problems and cost-driving factors in the software industry (approx. 50 % of total IT-cost, 80% of total software cost in integration issues …)
- There is an increasing need to systematically address integration in accidental information infrastructures and IT-architectures, that have grown over time in an uncontrolled manner in heterogeneous enterprise environments
- Integration needs:
  - Data and information integration
  - Software integration (full interoperability)

Model-based SWE / SDI : lots of promises

- Model-based development helps in sustainability (models are more stable than code …, abstraction level of documentation, …)
- Model-based integration saves approx. 70% of integration costs (statements by Bran Selic IBM/Rational, experiences from BIZYCLE experiments)
- Fully model-based approach (including process mgmt, and generated code/ model execution) can save even more than 90% of the total development cost

Cost impact (slide by András Pataricza, MBSDI 2009, Sydney)



AGENDA of the eBISS Tutorial on MBSDI

- History & Background
- Goal: Model Based Software & Data Integration

- **Languages and Models**
- **Metalanguages and Metamodels**

- BIZYCLE: Model Based Software & Data Integration

- BIZWARE: Domain Specific Languages

What is a Model? – Some Definitions...

(for a more general view, please, refer to: “Models in Science” ,Roman Frigg and Stephan Hartmann (2006), Stanford Encyclopedia of Philosophy http://plato.stanford.edu/entries/models-science/#RepModIIModDat)

- Merriam Webster dictionary:
  - 4.) a (usually) *miniature representation of* something; also : *a pattern of* something to be made
  - 10.) a system of *postulates, data, and inferences* presented *as a mathematical description of* an entity or state of affairs

- Grady Booch: ... *a simplification of reality* ...

- According to Stachowiak [Herbert Stachowiak, Allgemeine Modelltheorie, Springer, 1973] a model needs to possess three features:
  - *mapping feature*: A model is based on an original.
  - *reduction feature*: A model only reflects a (relevant) selection of the original's properties.
  - *pragmatic feature*: A model needs to be usable in place of the original with respect to some purpose.

Model Theory: Steinmüller

A model is information ...

- ... on something (content, meaning)
- ... created by someone (sender)
- ... for somebody (receiver)
- ... for some purpose (usage context)

translated from: W. Steinmüller. Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.

Abstraction is the key-concept to build models

- ... derive information from different viewpoints
- ... derive the essence, the characteristics, the lawfulness (‚Gesetzmäßigkeit‘) from a set of different individuals
- ... make relationships between concepts visible by deleting details

Examples of Models

- Mathematical Models

e.g. CIRCLE (*an idealized set of points in two-dimensional space*)

- Architectural Models

e.g. KING‘S CASTLE (*an idealized wish of an old/new building in the center of Berlin*)

- Technical Models

e.g. a prototype of a new car, but also all previous technical sketches, specifications, calculations, simulations, ... (*an idealized imagination of several communities*)

In our setting of Software and Information Systems:

- Models of the real world in order to achieve software solutions with given properties:
  - Mapping of the reality
  - Abstraction from (too many) details of the reality: reduction/ simplification
  - Pragmatics/ Feasibility

Examples: Mathematical Model of a Circle

- In general a set of mathematical formulae, visualized by some graphical drawing ...
- Coordinates equation ...
  ... all peripheral points of the circle with midpoint M and radius r :

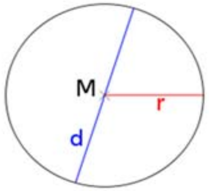
$$(x - x_M)^2 + (y - y_M)^2 = r^2$$

- Parameter representation (polar coordinates) with $0 \leq \varphi < 2\pi$:

$$x = x_M + r \cos \varphi$$

$$y = y_M + r \sin \varphi$$

- Graphical visualization of a circle with midpoint M, radius r and diameter d :



Architectural Model: KINGS CASTLE in Berlin

- In general miniature constructions in scale 1:100 ...

- Source: Google/Berliner Schloss Modell 6984.jpeg (visited Oct 22, 2012)

Technical Model: Volkswagen up!

- Prototype photo by Volkswagen, advertizing their new mini car: „up!“ ...
  ... for reasons of space we hide the approx. 500 Gbyte of technical models, specifications, calculations, ... ☺

What is Language?

- A **language** is a *dynamic set* of visual, auditory, or tactile *symbols of communication* and the elements used to manipulate them.
- A *set of commonly accepted symbols* is only one feature of language; all languages must define the structural relationships between these symbols in a *system of grammar*. Rules of grammar are one of the characteristics sometimes said to distinguish language from other forms of communication. They allow a finite set of symbols to be manipulated to create a potentially infinite number of grammatical utterances.
- Languages can be (among other classifications) subdivided into:
  - Formal language / Artificial language, mathematical and other languages created for a specific purpose, like e.g. first order logic (formal) or modeling/programming languages (artificial /semiformal)
  - Natural language, a language used naturally by humans

(A collection of statements from Wikipedia, modified and shortened according to my intensions)

Syntax / Semantics / Pragmatics (I)

In the linguistics of both natural and computer languages, the terms *syntax, semantics and pragmatics* are used to categorize descriptions of language characteristics.

- **Syntax :** The *syntax* of a language describes the structure and composition of allowable phrases and sentences of the language.
- **Semantics :** But syntax itself is devoid of *meaning*, simply telling us what strings are valid and how they may be parsed or decomposed. The meaning of these syntactic elements must be provided through *semantics.* In essence, we may think of providing syntactic elements as inputs to a *semantic function*, which in turn provides some representation of the meaning of the elements as output.
- **Pragmatics :** *Pragmatics* is the third general area of language description, referring to practical aspects of how constructs and features of a language may be used to achieve various objectives.

- From [Cameron 2002] **Robert D. Cameron, "Four Concepts in Programming Language Description: Syntax, Semantics, Pragmatics, and Metalanguage"**, Univ. San Francisco CA, 2002

Syntax / Semantics / Pragmatics (II)

- *Syntax:* Syntax is the study that relates signs to one another.

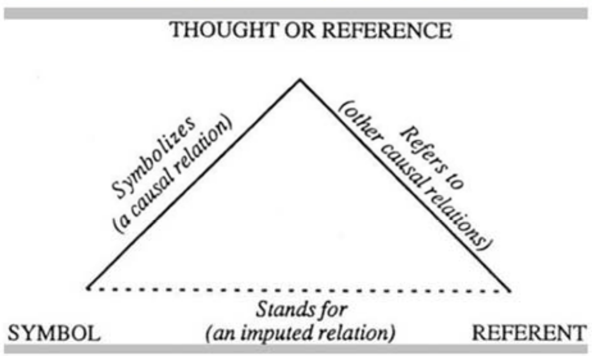
- *Semantics:* Semantics is the study that relates signs to things in the world and patterns of signs to corresponding patterns that occur among the things the signs refer to.

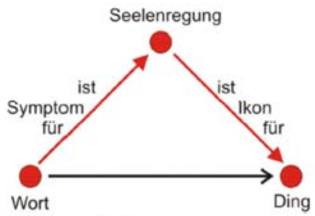
- *Pragmatics:* Pragmatics is the study that relates signs to the agents who use them to refer to things in the world and to communicate their intentions about those things to other agents who may have similar or different intentions concerning the same or different things.

- From [Sowa 2000] **John F. Sowa, "Ontology, Metadata, and Semiotics"** , in: B. Ganter & G. W. Mineau, eds., *Conceptual Structures: Logical, Linguistic, and Computational Issues, Lecture* Notes in AI #1867, Springer-Verlag, Berlin, 2000, pp. 55-81.

Meaning/Semantics – Bedeutung/Semantik

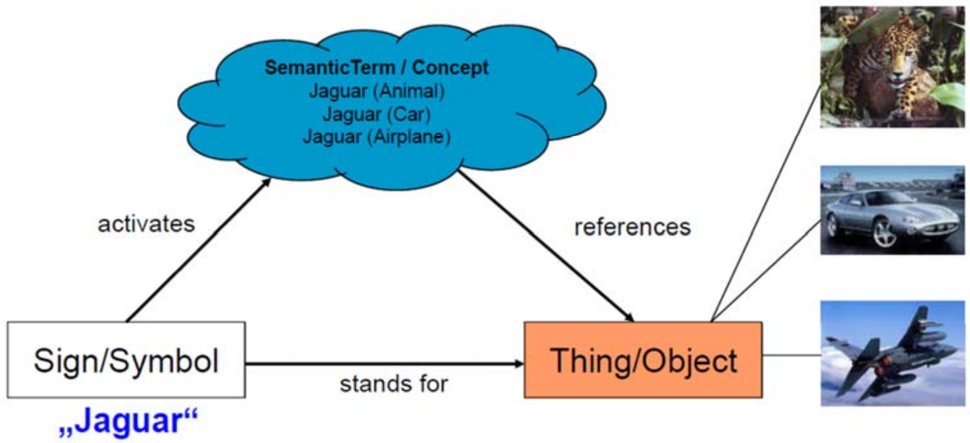
The Ogden and Richards (1923) Semiotic Triangle:





… and its predecessor … (Aristotle, 4th century BC, from Wikipedia)

Semiotic Triangle: Example



Semiotic Triangle (... some details ...)

- Ogden & Richard's (1923) famous triangle of meaning implies that the referent of an expression (a word or another sign or symbol) is relative to different language users.

- With the terminology of Peirce: "A sign, or *representamen*, is something which stands to somebody for something in some respect or capacity. It addresses somebody, that is, creates in the mind of that person an equivalent sign, or perhaps a more developed sign. That sign which it creates I call the *interpretant* of the first sign. The sign stands for something, its object [or referent]. It stands for that object, not in all respects, but in reference to a sort of idea, which I have sometimes called the *ground* of the representamen." (Peirce, 1931-1958, vol. 2, p. 228).

Semiotics

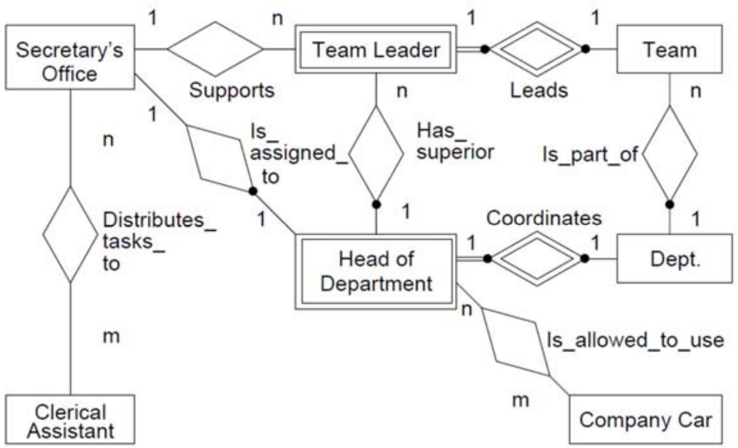
- Sowa (2000) writes about Ogden & Richards' (1923) triangle of meaning: "The triangle has a long history. Aristotle distinguished objects, the words that refer to them, and the corresponding experiences in the *psyché.* Frege and Peirce adopted that three-way distinction from Aristotle and used it as the semantic foundation for their systems of logic. Frege's terms for the three vertices of the triangle were *Zeichen (sign) for the symbol, Sinn* (sense) for the *concept*, and *Bedeutung* (reference) for the *object*."

Literature:

- Ogden, C. K. and I. A. Richards, I. A. (1923). *The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism.* London: Routledge & Kegan Paul.
- Peirce, C. S. (1931-1958). *Collected Papers of C. S. Peirce* ed. by C. Hartshorne, P. Weiss, & A. Burks, 8 vols., Harvard University Press, Cambridge, MA.
- Sowa, J. F. (2000). Ontology, Metadata, and Semiotics. In: B. Ganter & G. W. Mineau, eds., *Conceptual Structures: Logical, Linguistic, and Computational Issues,* LNAI 1867, Springer, Berlin, 2000, pp. 55-81.

Example: Information Modeling by E/R Diagrams

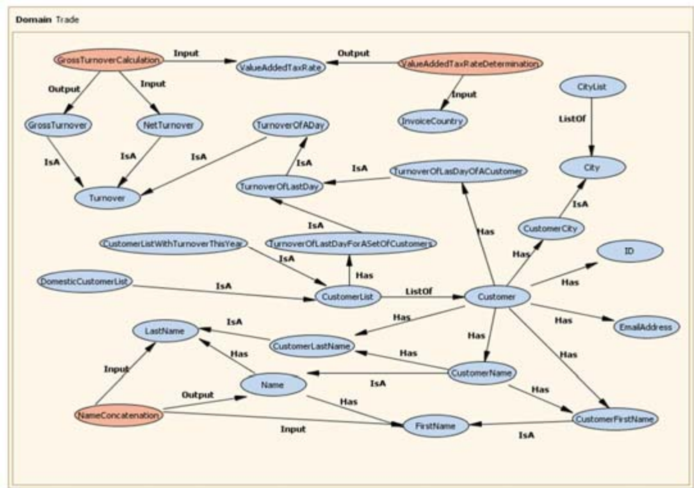
- Example: Excerpt of a phase 1-2 entity-relationship-diagram on some company structure aspects



(Source:
Lab example from an INFMOD course by Ralf Kutsche, 2003)

Example: Modeling „Knowledge“

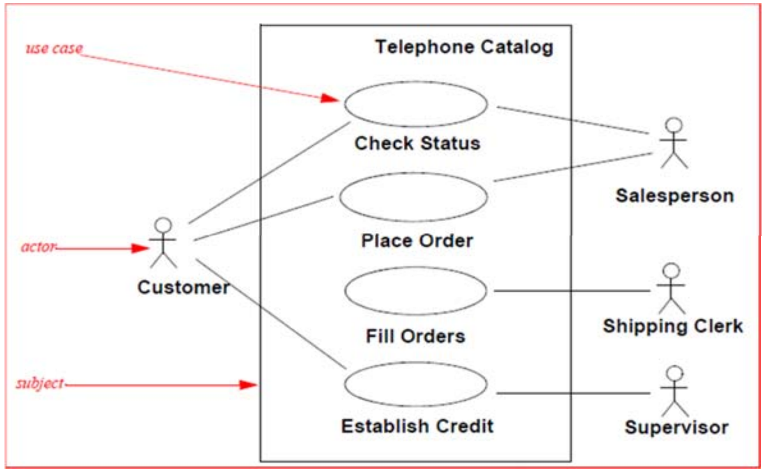
- Example:
  Customer Relationship Management (CRM) Ontology
  (starting point, finally very complex with lots of logical constraints in the background ...)



(Source:
BIZYCLE Project,
TU Berlin, 2009)

Software Modeling: System Functionality

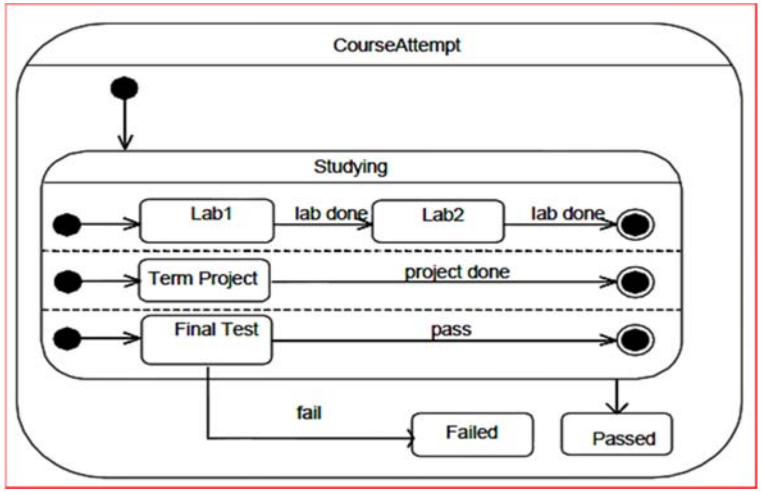
- Example: Modeling (coarse) functionality of a system for telephone orders (*UML use case diagram*)



(from: UML Superstructure Specification v2.3, OMG document formal/2010-05-05, 2010)

Software Models: Statechart Diagram

- Example: Modeling the (concurrent) states of a student during a university project course (*UML statecharts*)



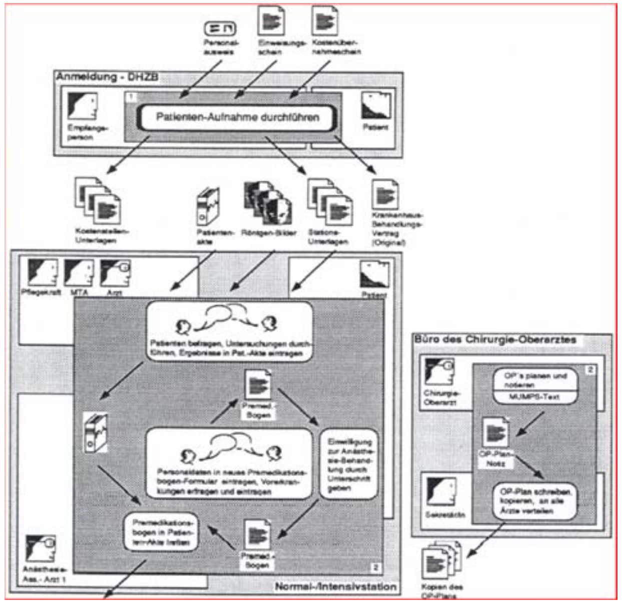
(from: UML Superstructure Specification v2.3, OMG document formal/2010-05-05, 2010)

Modeling example: Activity & Document Flows

- Health Care Example:

Combined Workflow and Document Flow Diagram including the dimensions of time, space (local org units), and responsibility

(Patient registration für heart surgery, excerpt)



(Source: Ralf-D. Kutsche, 1990, Deutsches Herzzentrum Berlin)

Metalanguage

- Another important concept that applies to all aspects of language description is that of **metalanguage**. *Metalanguage*, in general, *refers to the language in which a subject language is being described*. For example, BNF (Backus-Naur Form) is a metalanguage widely used to describe the syntax of programming languages. Similarly, there are formal metalanguages for describing the semantics of programming languages, particularly associated with the approaches of axiomatic semantics and denotational semantics. In most cases, however, a less formal approach to semantic description is taken, using English as the metalanguage.

- It is important to note that metalanguages are indeed languages in their own right. In particular, one should expect that metalanguages each have their own syntax, semantics and pragmatics, which in turn must be described by a **metametalanguage**. Typically, a combination of English prose and standard mathematical notation is used as metametalanguage.

- From [Cameron 2002] **Robert D. Cameron, "Four Concepts in Programming Language Description: Syntax Semantics Pragmatics and Metalanguage"**, Univ. San Francisco CA, 2002

Repetition: UML main modeling languages

In order to allow for multi view modeling, UML provides (among others, particularly in UML 2.x) the following kinds of models/diagrams:

- use case diagram
- class diagram
- behavior diagrams:
  - statechart diagram
  - activity diagram
- interaction diagrams:
  - sequence diagram
  - collaboration diagram
- implementation diagrams:
  - component diagram
  - deployment diagram

Basic Philosophy of UML

The selection of models and diagrams used for modeling a concrete development (software, information, embedded, ...) system, massively influences the development process and the solution. Abstraction, i.e. to concentrate on only relevant parts, models and details, while neglecting others, is one key to success, multi view modeling the other: (*citations from [UML 03]*)

- "Every complex system is best approached through a small set of nearly independent views of a model."
- "No single view is sufficient."
- "Every model may be expressed at different levels of fidelity."
- "The best models are connected to reality."

Basic Philosophy of UML (cont'd)

The whole set of UML diagrams for a given problem provides many perspectives (views) on a system in analysis, design and development. The underlying UML metamodel, describing the modeling concepts and their relationships, allows for the integration of these perspectives. The resulting diagrams, taken together with all necessary documentation (requirement texts, glossary of terms, logical constraints, design decisions), are the essential artefacts for a model-driven development process.

Keywords (for own context literature discovery):

- Model-driven architecture MDA, adressing UML, MOF and CWM in its kernel (by Object Management Group OMG, www.omg.org/mda)
- Multi view modeling
- Multi perspective software development
- Open Distributed Processing (RM-ODP, www.rm-odp.net)

Models and Metamodels

The required **integrated and unified (multi-perspective) view on (software, information, embedded,...) systems** – in our case: complex heterogeneous information systems – can only be achieved, if the language constructs of the different (part) languages used for modeling and providing the **different perspectives and views, are connected via an abstract higher level model, the so-called metamodel**.

This can be achieved by metamodel hierarchies like the MOF[1]- and UML-**metamodel-hierarchy.**

Def. Metamodel (in UML context ...)

A metamodel is a model which describes precisely (semi-formally) the concepts of a modeling language, in order to let (syntactically) conform the well-formedness of a specification language. (from [UML 03])

1. MOF — Meta Object Facility, a specification of the Object Management Group OMG, in order to enable the description of modeling languages and their constructs on an abstract basis, such that integration, transformation and exchange of models in different languages is possible. In the following cited as [MOF 02]: Meta Object Facility (MOF) Specification, v1.4, April 2002, formal/2002-04-03.

Metamodel: Definition

OMG MDA Guide

- „[A metamodel is] a model of models“

OMG UML 2.0 Infrastructure specification

- „A model in an instance of a metamodel“
- A metamodel is the collection of "concepts" (aka things, terms, ...) that are the vocabulary with which you are talking about a certain domain.

Abstraction is the key-concept to build metamodels

- ... derive information from different viewpoints
- ... derive the essence, the characteristics, the lawfulness (‚Gesetzmäßigkeit‘) from a set of different individuals
- ... make relationships between concepts visible by deleting details

Language and Metamodeling / the UML example

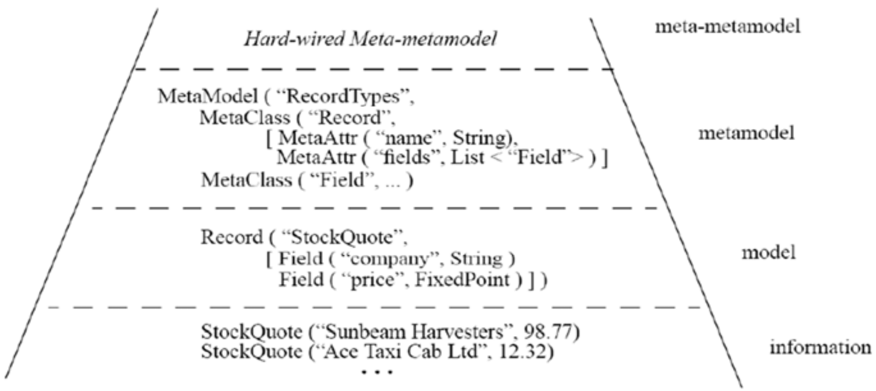
- In order to explain briefly the concept of metamodeling, we use for the introduction the (older, but much more intuitive) UML v1.5 metamodel, provided by the Object Management Group OMG: *Unified Modeling Language, v1.5, March 2003, formal/03-03-01,* in the following always shortly cited as [UML 03].

- A comprehensive and integrating view of the modeling language family of UML is given by its **metamodel**, consisting of
  - the **abstract syntax**, given by UML class diagrams (*concepts*)
  - **rules and conditions for well-formedness of UML diagrams in natural language and in OCL**[1]
  - a very comprehensive **documentation of all basic concepts of UML in natural language**, in order to describe the **meaning of all UML language constructs (*semantics*).**

1. OCL — *Object Constraint Language, i.e. the logical language (essentially first order logic) of UML, in order to specify conditions, dependencies, and rules.*

MOF Metamodel Hierarchy

The MOF metamodel hierarchy[1] consists of 4 levels (’layers’):



1. taken from [MOF 02] Meta Object Facility (MOF) Specification, v1.4, April 2002, ch. 2.2.1, fig. 2-1, p. 2-2

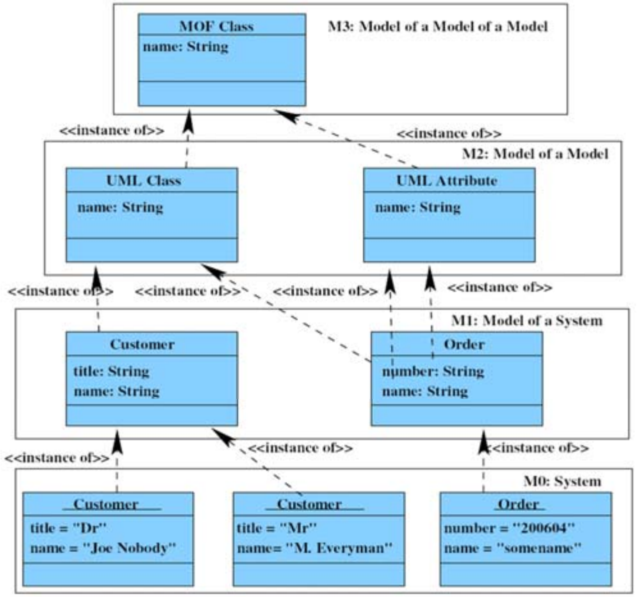
UML Metamodel Hierarchy

The UML metamodel hierarchy[1] also shows 4 levels:

Layer	Description	Example
meta-metamodel	The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.	MetaClass, MetaAttribute, MetaOperation
metamodel	An instance of a meta-metamodel. Defines the language for specifying a model.	Class, Attribute, Operation, Component
model	An instance of a metamodel. Defines a language to describe an information domain.	StockShare, askPrice, sellLimitOrder, StockQuoteServer
user objects (user data)	An instance of a model. Defines a specific information domain.	<Acme_SW_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>

1. taken from [UML 03], ch. 2.2.1, p. 2-5

Classical MOF Layers



UML / MOF Metamodel Hierarchies – Summary

Model Level M3 — Meta Meta Model(s)

... description of the (abstract syntax and semantics of) meta model concepts (in case of MOF fixed): which meta model concepts? how to interrelate and to connect them?

Model Level M2 — Meta Models

... description of abstract syntax and of semantics of the modelling language: which model language constructs?, in which way interrelated and connected?

Model Level M1 — Models / Types

... the first conceptual level of modeling: concrete types / classes of objects and relationships, typed attributes, constraints, etc.

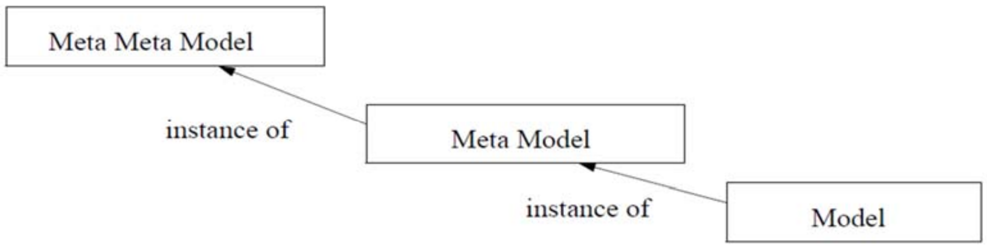
Model Level M0 — Data

... concrete object instances, data, attribute values, ...

Metamodel Hierarchies in General

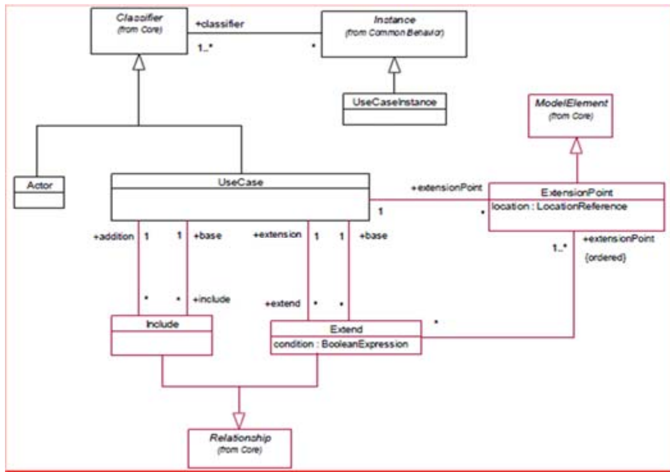
There is no theoretical reason to limit the number of layers in a metamodeling approach to just 4 — we could have arbitrary many, or even an infinite number of layers. However, for the pragmatics of building modeling tools and metamodeling tools, there is given a fixed number of – in general – 4 layers. The basic relation among the layers for this kind of metamodel appoach is given by ’instance of’:

- "layer (n-1) (meta) model **is instance of** layer n (meta) metamodel“



Language Definition: Example Use Case M.M.

- Repetition: A Metamodel describes the *abstract syntax of a modeling language*, i.e. its concepts, i.e. Use Case Models:



(from: OMG UML Specification v1.5, OMG document formal/03-03-01, March 2003)

UML Metamodel: Core

The ’core package’ defines the basic abstract and concrete constructs (and their semantics) in the UML metamodel needed for the construction of all UML languages:

Abstract (i.e. non-instantiable) constructs serve for organizing and structuring the UML metamodel, like ’ModelElement’, ’GeneralizableElement’ or ’Classifier’.

Concrete UML metamodel constructs are instantiable and reflect the elements being used in the UML modeling, like, e.g.,’Class’, ’Attribute’, ’Operation’ or ’Association’.

Core Package: Backbone

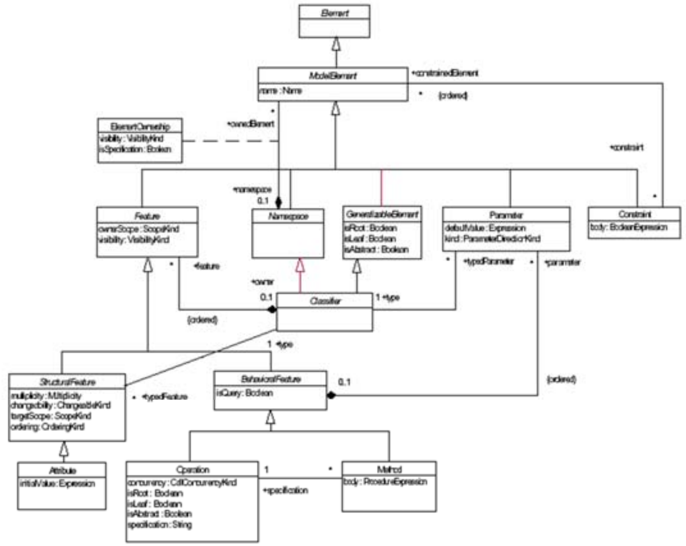


Figure 2-5 Core Package - Backbone

Core Package: Relationships

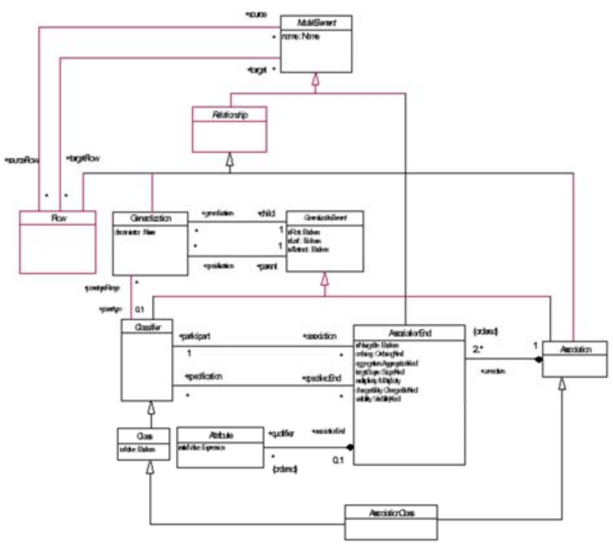


Figure 2-6 Core Package - Relationships

Core Package: Classifiers

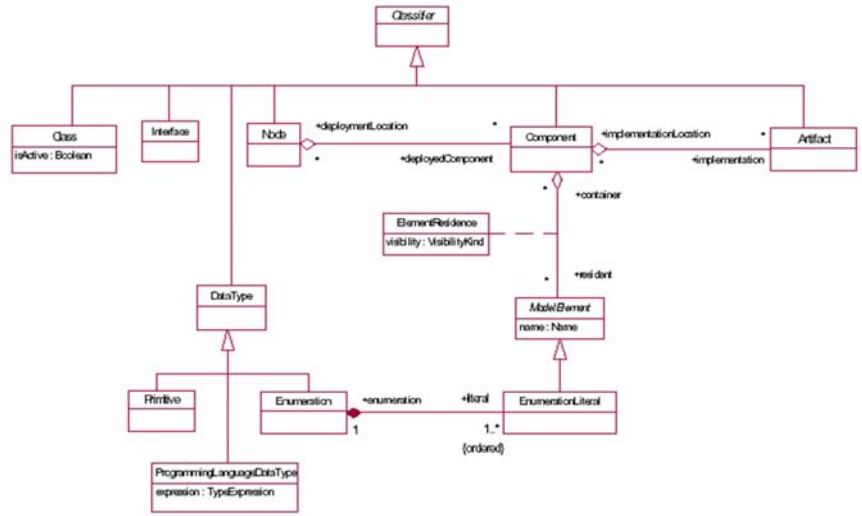
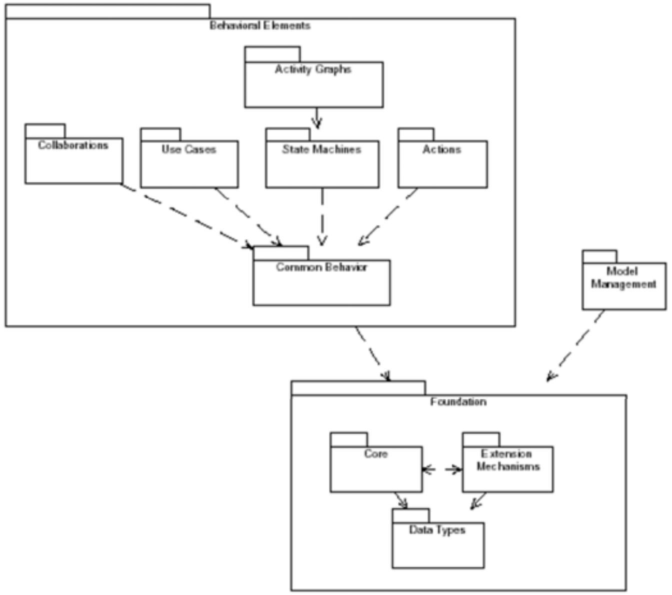


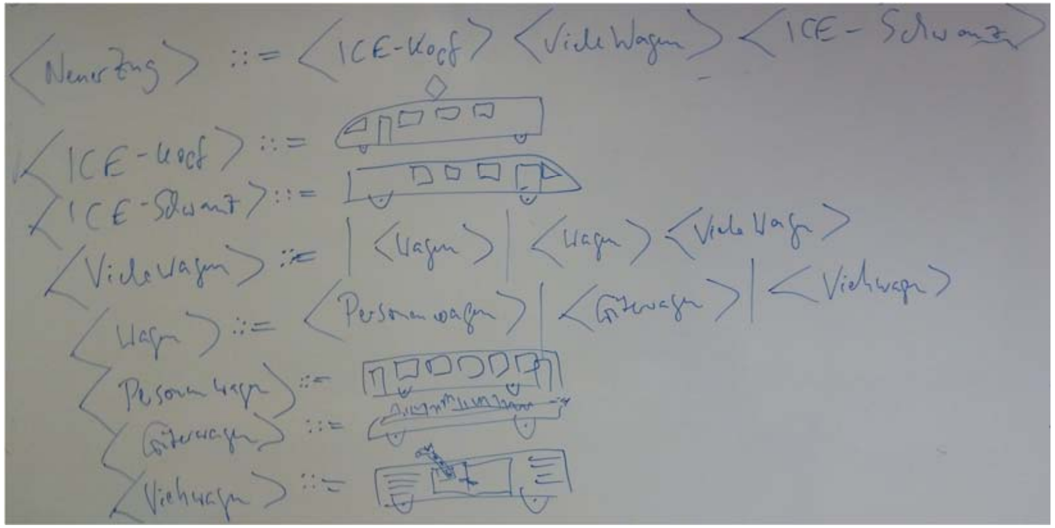
Figure 2-8 Core Package - Classifiers

Overview: Package-Structure of the UML 1.5 Metamodel



Language Definition: another example …

- … BNF (Backus-Naur-Form), here mixing textual and graphical elements!



(Source: an industrial training by Ralf Kutsche, 2011)

Further Reading in Metamodeling

- **[1] Colin Atkinson & Thomas Kühne.** *Model-driven development: A Metamodeling Foundation*, IEEE Software, Vol. 20, No. 5, pp. 36-41, 2003
- **[2] Thomas Kühne.** *Matters of (Meta-) Modeling*, Journal on Software and Systems Modeling, Vol. 5, No. 4, pp. 369-385, Dec 2006
- **[3] Jean Bezivin and Olivier Gerbe.** *Towards a precise definition of the OMG/MDA framework*, Proc 16th Int. Conf. on Automated Software Engineering, Coronado Island, pages 273–280, November 2001
- **[4] Jean Bezivin**. *In search of a basic principle for model driven engineering*, Special Novatica Issue “UML and Model Engineering”, V(2), April 2004
- **[5] Susanne Strahringer,** *Metamodellierung als Instrument des Methodenvergleichs,* Shaker Verlag, Aachen, 1996

UML 2 Discussion of MOF levels

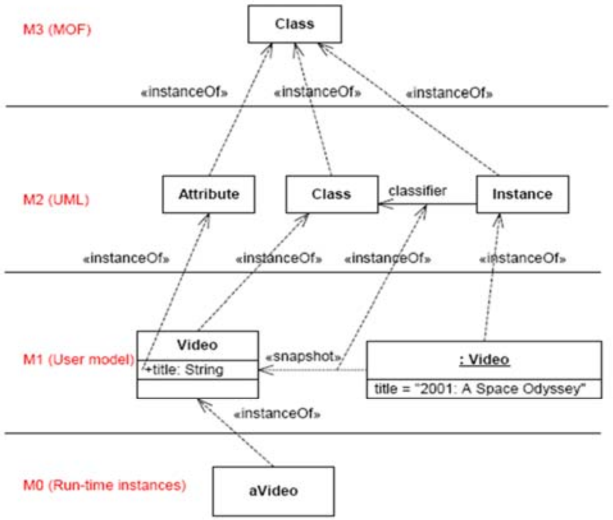
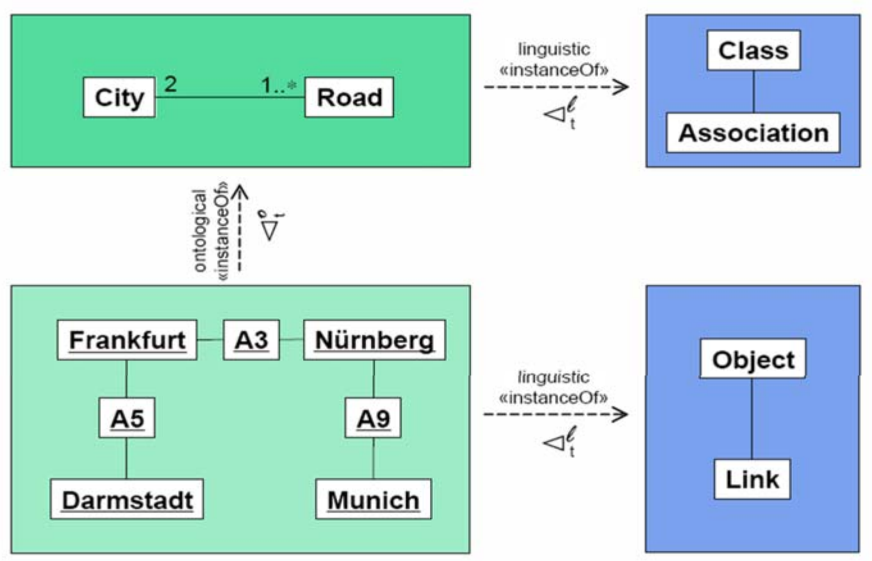


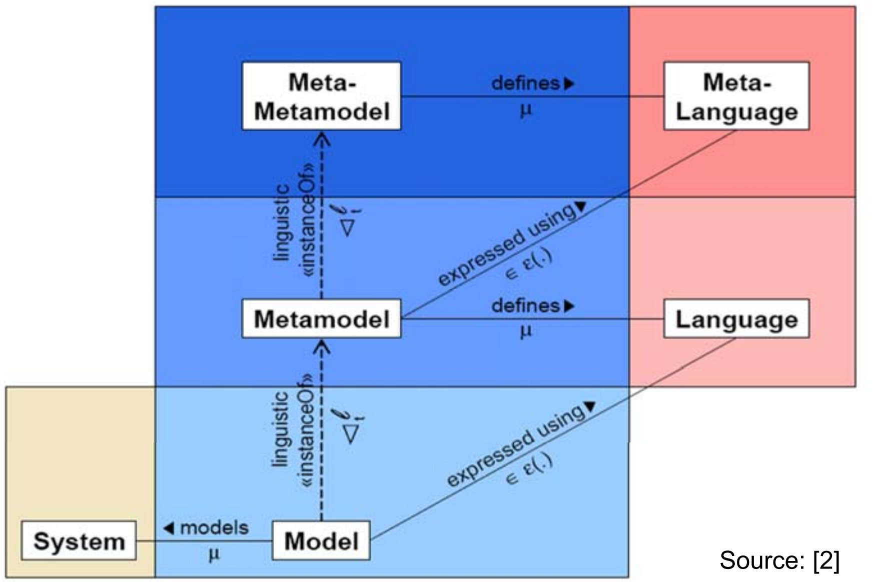
Figure 7.8 - An example of the four-layer metamodel hierarchy

Ontological/ Linguistic Metamodels



Source: [2]

(Meta-)(Meta-)Models and (Meta-)Languages



Source: [2]

Notations of Kühnes ‚Matters of M.M.'

notation	*name*	*description*
α	abstraction	creates a model from a system using projection (π) and possibly classification (Λ) and generalization (Γ), hence $\mathcal{S} \lhd \alpha(\mathcal{S})$.
Λ	classification	creates a type model, hence $\mathcal{M} \lhd_t \Lambda(\mathcal{M})$
Γ	generalization	creates a supermodel, hence $\mathcal{S} \lhd_t \mathcal{M} \to \mathcal{S} \lhd_t \Gamma(\mathcal{M})$
π	projection	a homomorphic mapping creating a reduced system from a given system, using selection and reduction of information.
ρ	represents	records the intention of a model to represent a system.
μ	meaning	assigns meaning to a model (element). If $\rho(\mathcal{S},\mathcal{M})$ then one may define $\mu(\mathcal{M}) = \pi(\mathcal{S})$.
$\lhd$	model-of	holds between a system and a model describing the former.
$\lhd_i$	token model-of	holds between a system and a model representing the former in a one-to-one fashion. Model elements may be regarded as designators for system elements.
$\lhd_t$	type model-of	holds between a system and a model classifying the former in a many-to-one fashion. Model elements are regarded as classifiers for system elements.
$\lhd^o$	ontological model-of	indicates that the model controls the *content* of its elements, hence $\mathcal{S} \lhd_t^o \mathcal{M} \rightleftharpoons \mu(\mathcal{S}) \in \varepsilon(\mu(\mathcal{M}))$ and $\mathcal{S} \lhd_i^o \mathcal{M} \rightleftharpoons \mu(\mathcal{M}) = \pi(\mu(\mathcal{S}))$. Assuming $\mu(\mathcal{S}) = \mathcal{S}$, for systems which do not model anything, we have $\mathcal{S} \lhd_i^o \mathcal{M} \rightleftharpoons \mu(\mathcal{M}) = \pi(\mathcal{S})$ and thus $\rho(\mathcal{S},\mathcal{M}) \to \mathcal{S} \lhd_i^o \mathcal{M}$ (see definition of μ above).
$\lhd^l$	linguistic model-of	indicates that the model controls the *form* of its elements. This automatically implies $\lhd_t^l$ and, hence $\mathcal{S} \lhd_t^l \mathcal{M} \rightleftharpoons \mathcal{S} \in \varepsilon(\mu(\mathcal{M}))$

Table 2 Notation Overview

Remark: Model Transformations in Software & Data Integration[1]

A very important use of the metamodeling methodology is the transformation of (meta) models on different abstraction levels: Following the basic philosophy of MDA (*Model Driven Architecture) of the OMG, we organize all the models (and their according* metamodels) on the layers of CIM (*Computation Independent Models, i.e., roughly* spoken, on the business level), PIM (Platform Independent Models) and PSM (Platform Specific Models, i.e. taking concrete implementation platforms like OMG CORBA, .NET, WebServices, etc. into account on a specific modeling level).

For software and data integration business, you always find a very heterogeneous infrastructure of software solutions, data storage and workflows around the business functions. In order to integrate over this heterogeneity, you can relate metamodels to each other. Moreover, you can apply (abstraction and refinement) model transformations, in order to solve integration conflicts on a higher abstraction level of PIM models, abstracting from platform specific features on the PSM level. Even only using PSM models (instead of just working on code level) simplifies integration significantly.

1. These techniques are a very important vehicle in our BIZYCLE research project, providing a platform for model-based software and data integration and interoperability. We shall see this approach in the subsequent slides.

AGENDA of the eBISS Tutorial on MBSDI

- History & Background
- Goal: Model Based Software & Data Integration

- Languages and Models
- Metalanguages and Metamodels

- **BIZYCLE: Model Based Software & Data Integration**

- BIZWARE: Domain Specific Languages

BIZYCLE @ TU Berlin 2007 – 2010

Methodology, Design & Development
of the Model-based Interoperability Platform MBIF
for Software and Data Integration

Our Project team:
Dr. Ralf-D. Kutsche, Dr. Nikola Milanovic
Henning Agt, Gregor Bauhoff, Timo Baum, Mario Cartsburg, Hatice Elmasgünes,
Daniel Kumpe, Michael Shtelma, Jürgen Widiker
+ Students …

WACHSTUMSKERNE UNTERNEHMEN REGION Die BMBF-Innovationsinitiative Neue Länder
Bundesministerium für Bildung und Forschung
BIZYCLE

Technische Universität Berlin (TU Berlin)
Computergestützte Informationssysteme CIS / Datenbanksysteme und Informationsmanagement DIMA

BIZYCLE Vision 2006

- Methodology & technology platform for enterprise-wide (even: cross-enterprise) integration of software solutions and components in business intelligence
- Transparent access to all business relevant applications
- (Semi-) Automatized integration of applications for various kinds of business process improvement
- Cross-system evaluation of enterprise relevant data (KPIs..)

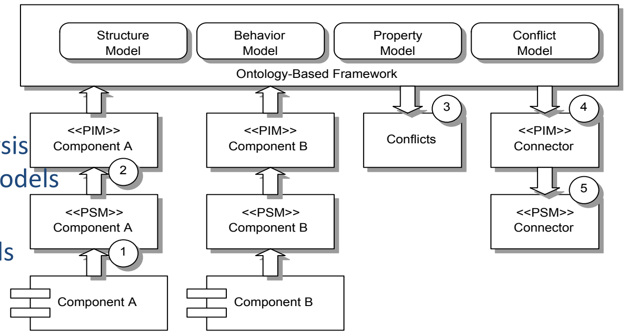
BIZYCLE – slogan:

Plug and Play Your Business!

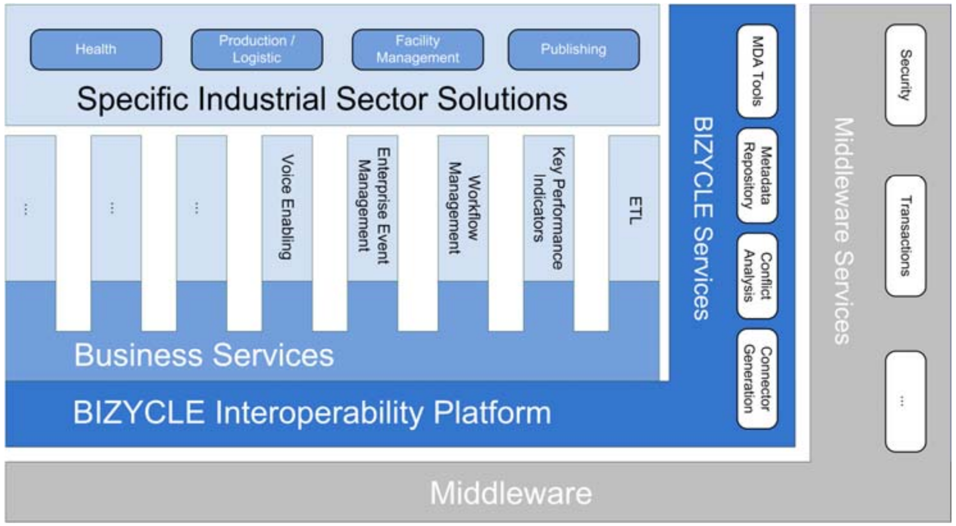
BIZYCLE Philosophy and Methodology 2006

- Integration scenarios are modeled at different abstraction levels:
  - Computation independent model level (CIM)
  - Platform independent model level (PIM)
  - Platform specific model level (PSM)

- Integration methodology:
  - Component/Interface Analysis Platform Specific (Meta-) Models
  - Transformation to Platform Independent (Meta-) Models
  - Conflict Analysis
  - Connector Generation



BIZYCLE Platform Architecture 2006



BIZYCLE Modeling Methodology & Metamodel Research for System & Component Integration

WACHSTUMSKERNE UNTERNEHMEN REGION Die BMBF-Innovationsinitiative Neue Länder

Bundesministerium für Bildung und Forschung

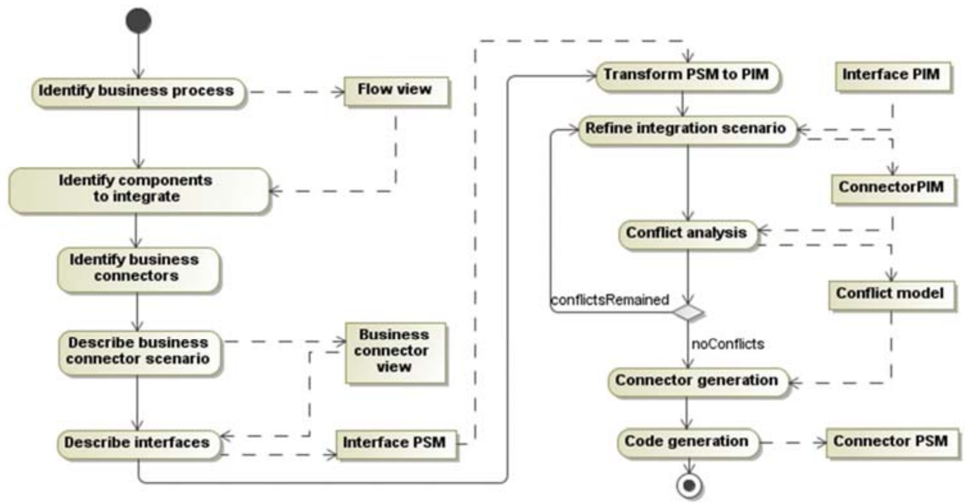
BIZYCLE

Technische Universität Berlin (TU Berlin)
Computergestützte Informationssysteme CIS / Datenbanksysteme und Informationsmanagement DIMA

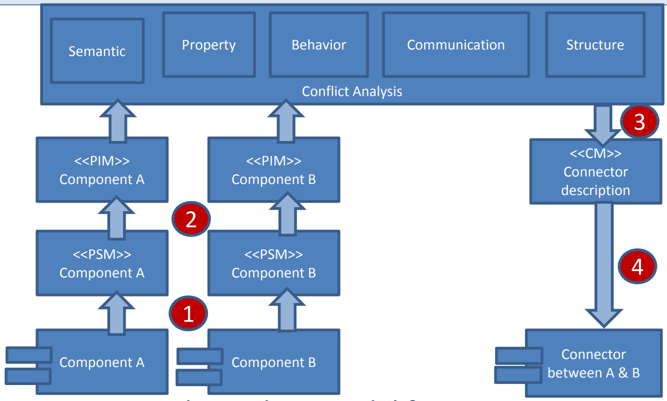
(Meta-)Modeling Methodology

- Methodology:
  Applying MDA methodology (CIM, PIM, PSM meta-/modeling)
- Application:
  Integration Scenario Modeling (CIM level)
  (starting from UML diagrams for use cases, interactions, activities, etc. … later using the MBIF toolsuite …)
- Research: Metamodel Development
- Application: Modeling Interface Descriptions / Metamodel Instantiation for Integration Scenarios (PSM level)

Component Integration Process 2007



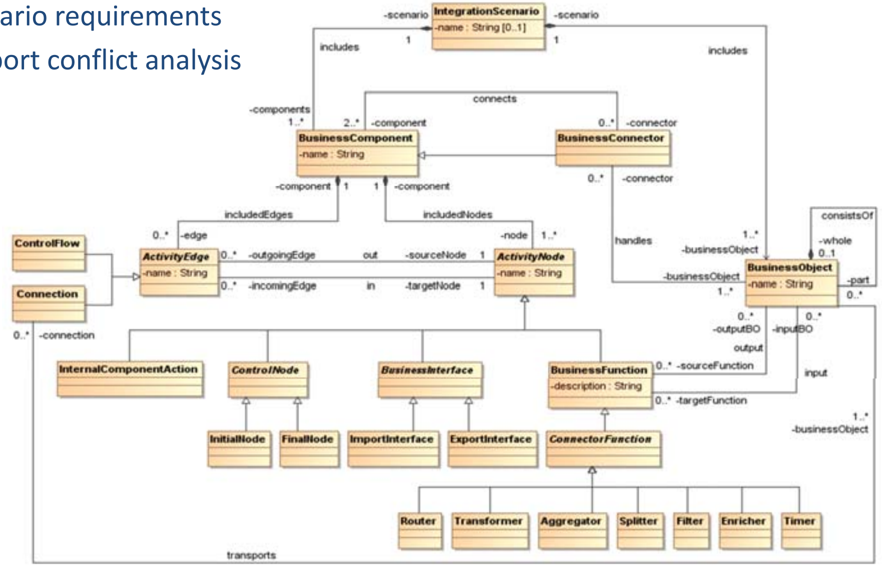
BIZYCLE (Meta-) Models



- CIM: Computation Independent Model for integration requirements
- PSM: Platform Specific Models for interface descriptions (many!)
- PIM: Platform Independent Model for interface descriptions
- SM: Semantic Model (domain ontology)
- AM: Annotation Model for semantic annotations
- CM: Connector Model

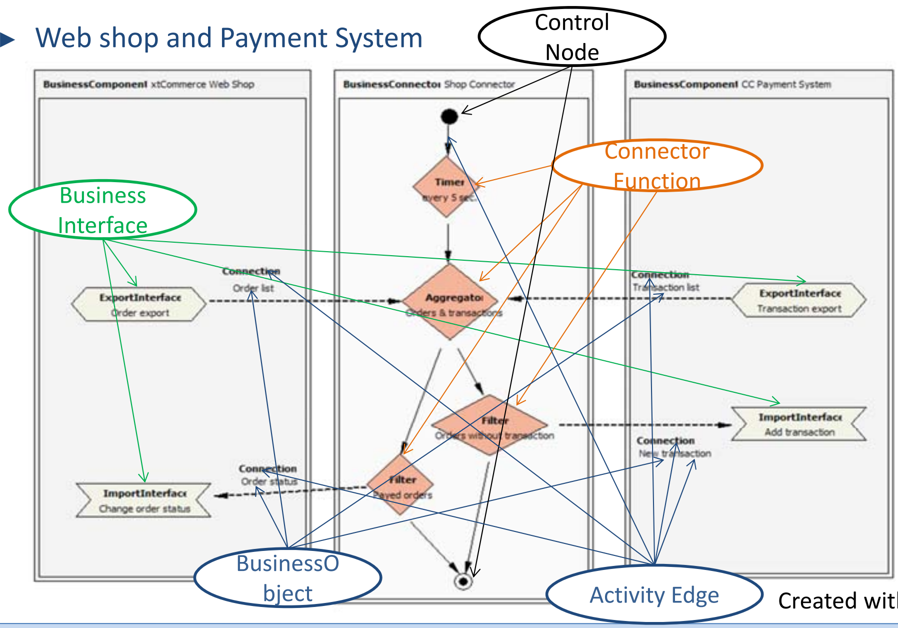
Computation Independent Metamodel (CIM)

- Captures business scenario with control and data flow
- Scenario requirements
- Support conflict analysis



CIM Example

- Web shop and Payment System



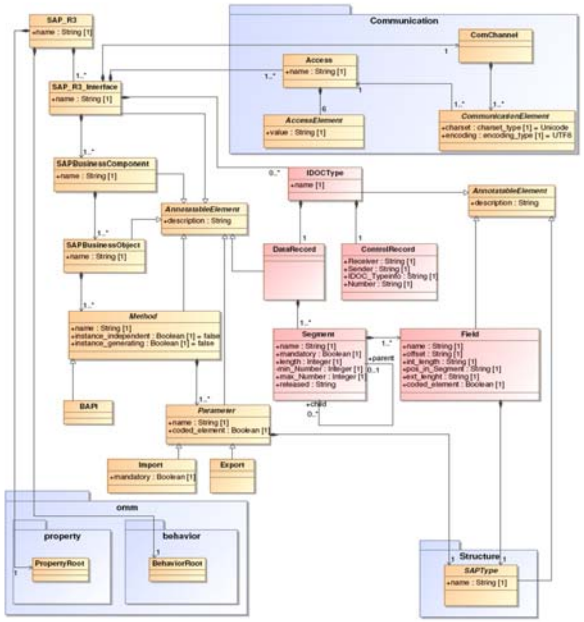
Created with MBIF

BIZYCLE – Modeling Interface Description

- Describe ‘technical’ information about interfaces that are to be integrated:
  - Interface static signature (types)
  - Interface *behavior*
  - *Communication* protocols
  - *Non-functional* properties
- Systems under study, i.e. development of PSMMs:
  - ERP (e.g., SAP R/3 BAPI/IDOC)
  - Relational and XML databases
  - J2EE and .NET components/applications
  - Web Services
  - Flat files (e.g., XML, CSV…)

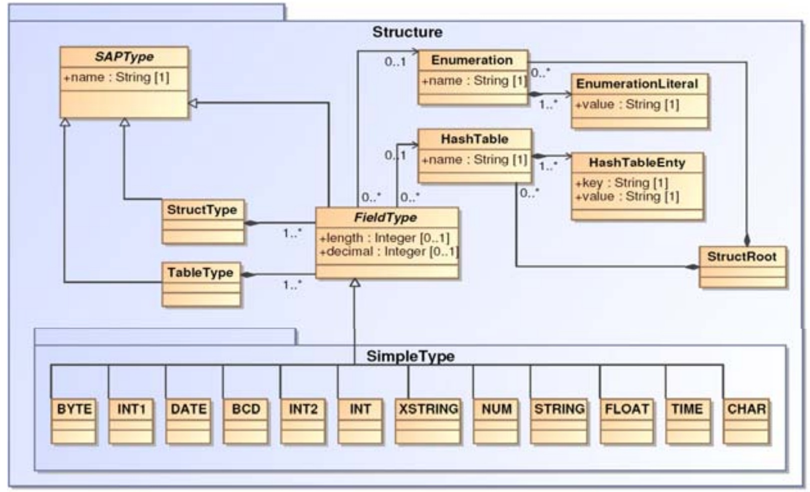
Platform Specific Metamodel for SAP

- Core



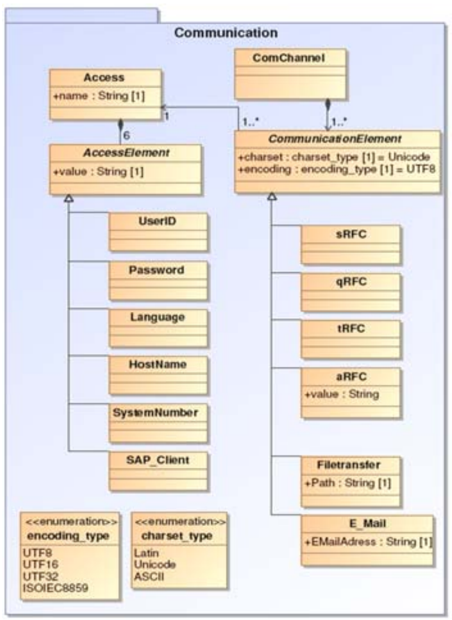
Platform Specific Metamodel for SAP

- Structure



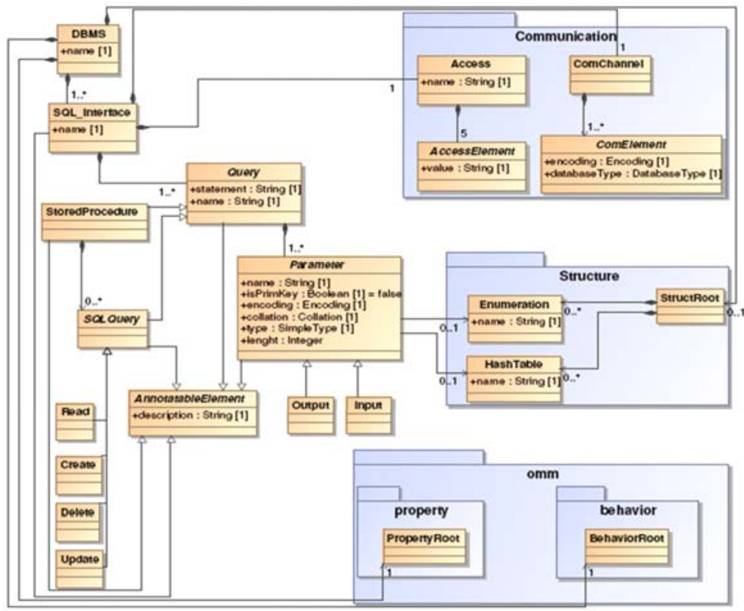
Platform Specific Metamodel for SAP

- Communication



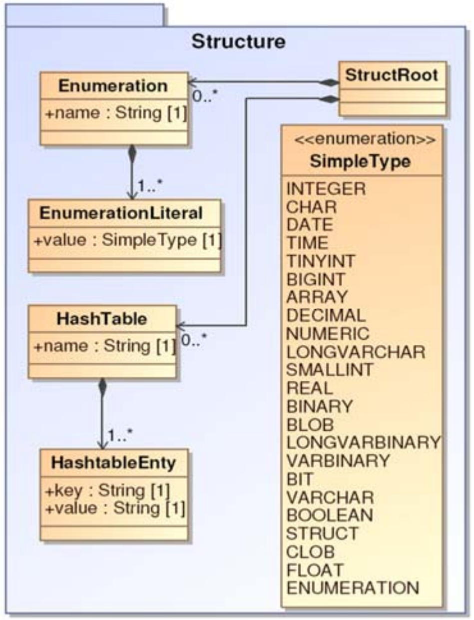
Platform Specific Metamodel for Rel.DB

- Core



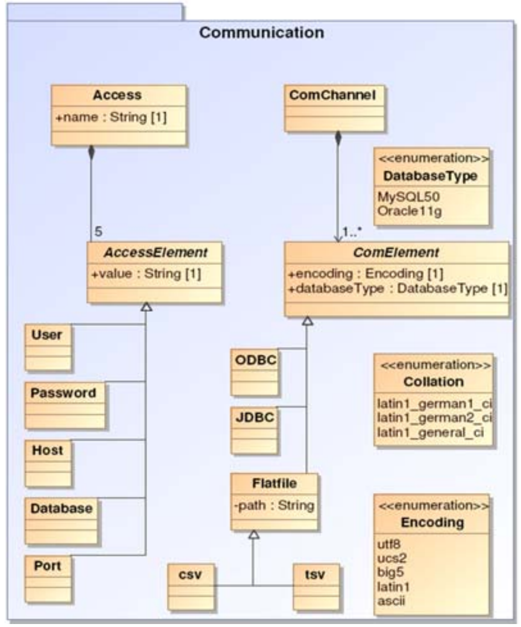
Platform Specific Metamodel for Rel.DB

- Structure



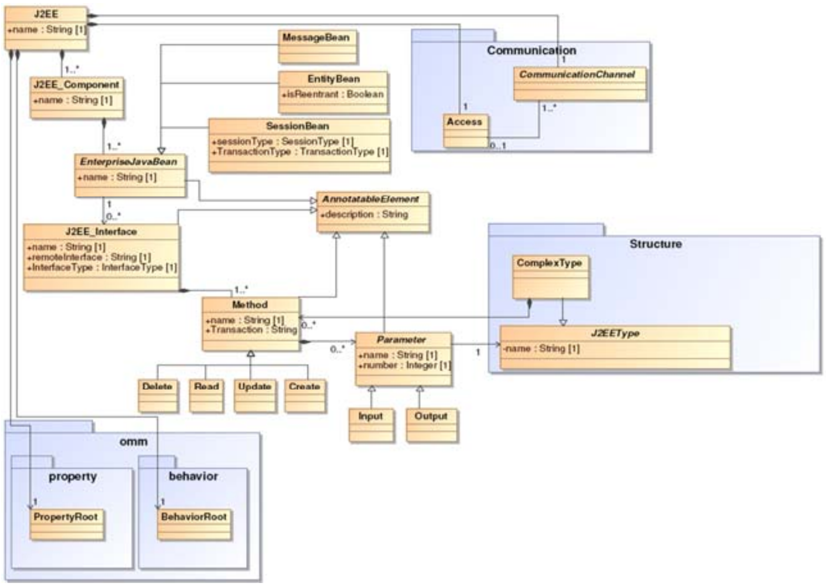
Platform Specific Metamodel for Rel.DB

- Communication



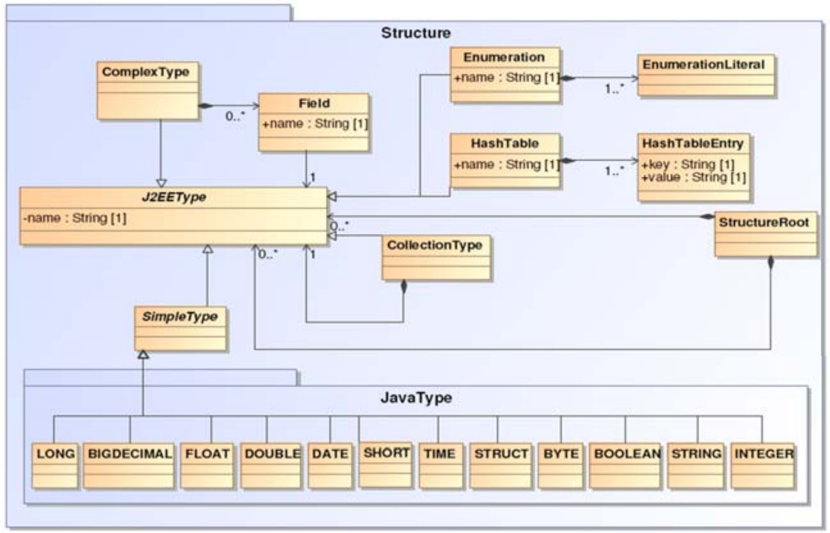
Platform Specific Metamodel for J2EE

- Core



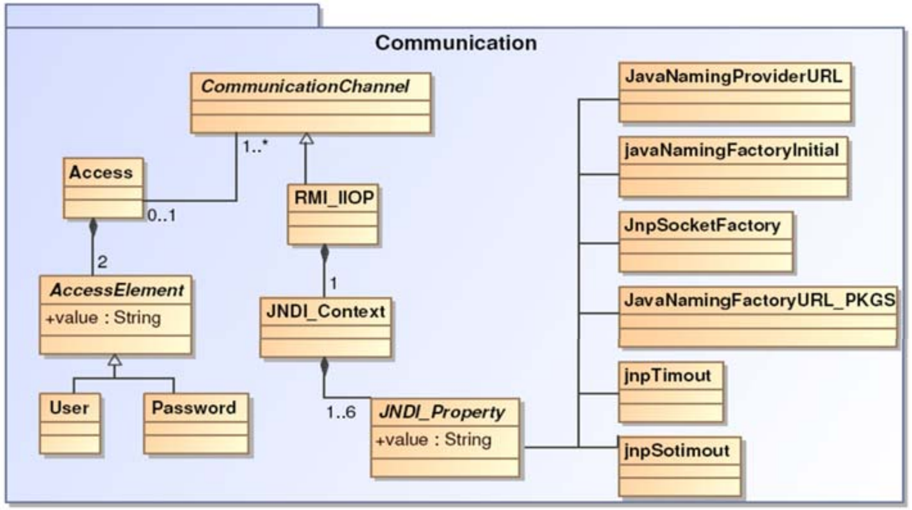
Platform Specific Metamodel for J2EE

- Structure



Platform Specific Metamodel for J2EE

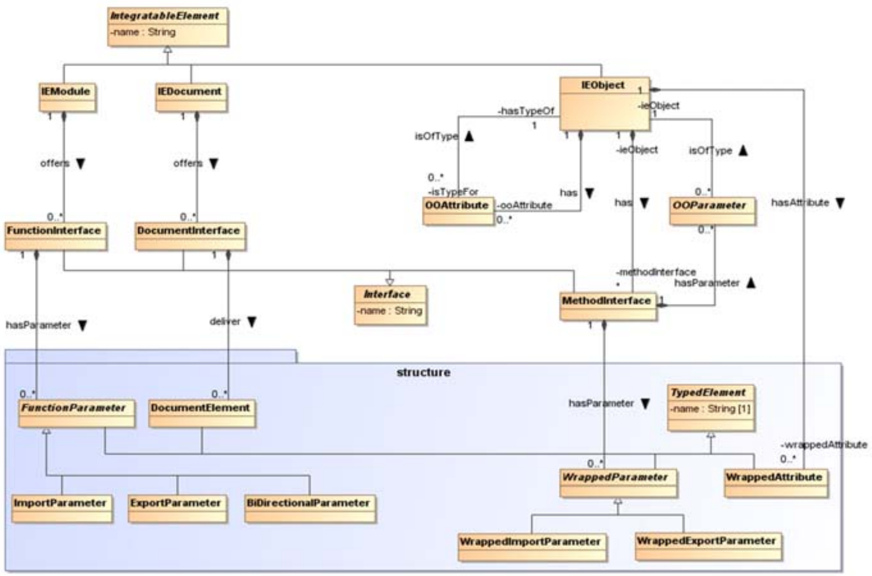
- Communication



Platform Independent Metamodel

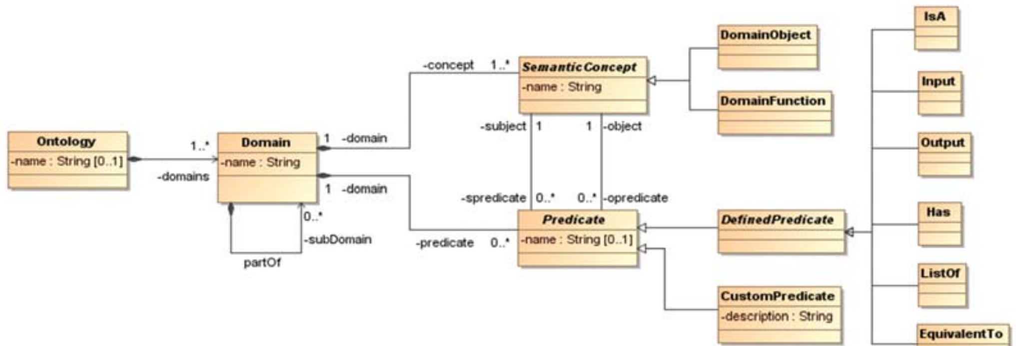
- PIM interface descriptions are generated from PSM interface descriptions by model transformation
- Abstract platform specific issues from PSM interface descriptions by generalizing interface properties
- Presents BIZYCLE middleware with a unified view of component interfaces
- Goal: to enable conflict analysis of heterogeneous components

Platform Independent Metamodel (Excerpt)



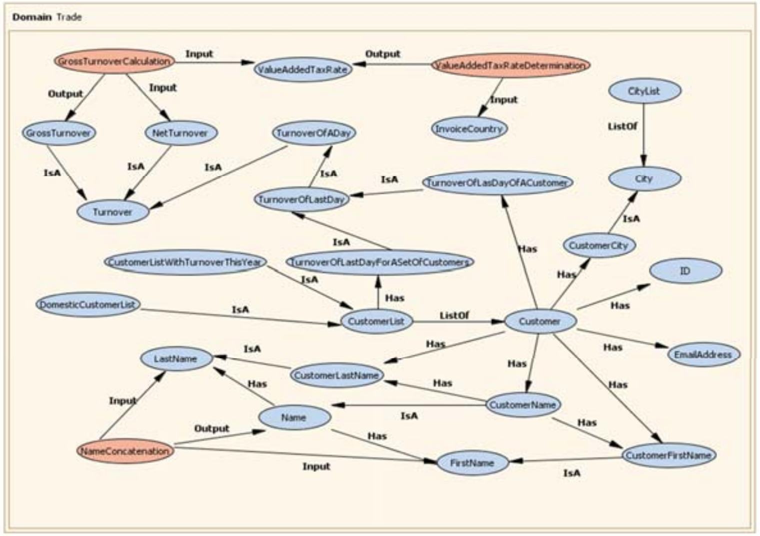
Semantic Metamodel

- Domain Ontology for Integration Scenario
- Controlled vocabulary
- Based on Resource Description Framework (RDF)
- *Subject – Predicate – Object* (RDF - triple)
- Needed for Annotations to PSM, PIM and CIM Elements



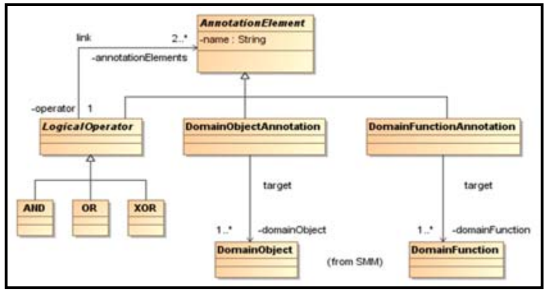
Semantic Model Example

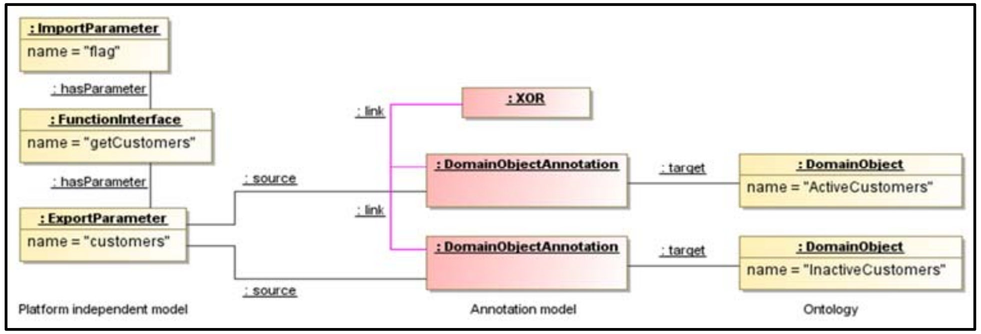
- Set of RDF triples: a *Trade* domain ontology



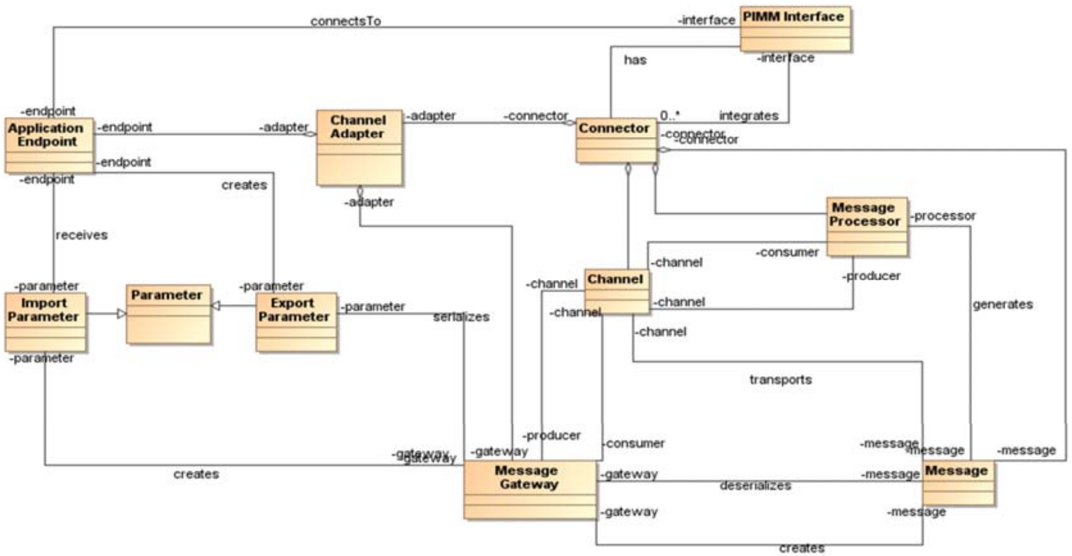
Annotation Metamodel

- Connects semantic to different models (CIM, PSM, PIM)

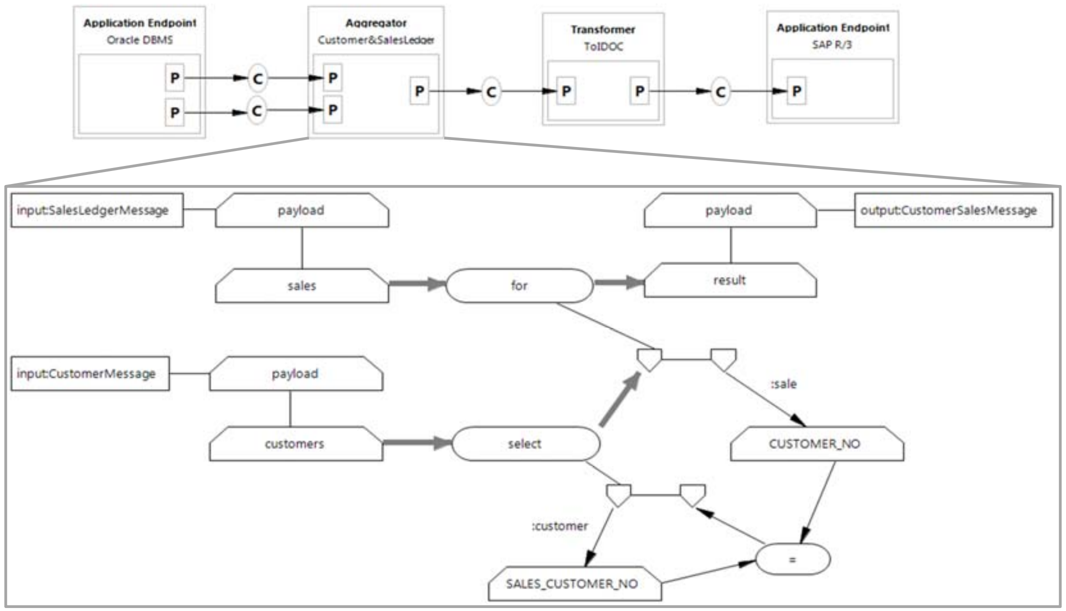




Connector Metamodel



Example Connector Model



BIZYCLE
Model&Metamodel, Metadata and Document Repository for Software and Data Integration

WACHSTUMSKERNE UNTERNEHMEN REGION – Die BMBF-Innovationsinitiative Neue Länder

Bundesministerium für Bildung und Forschung

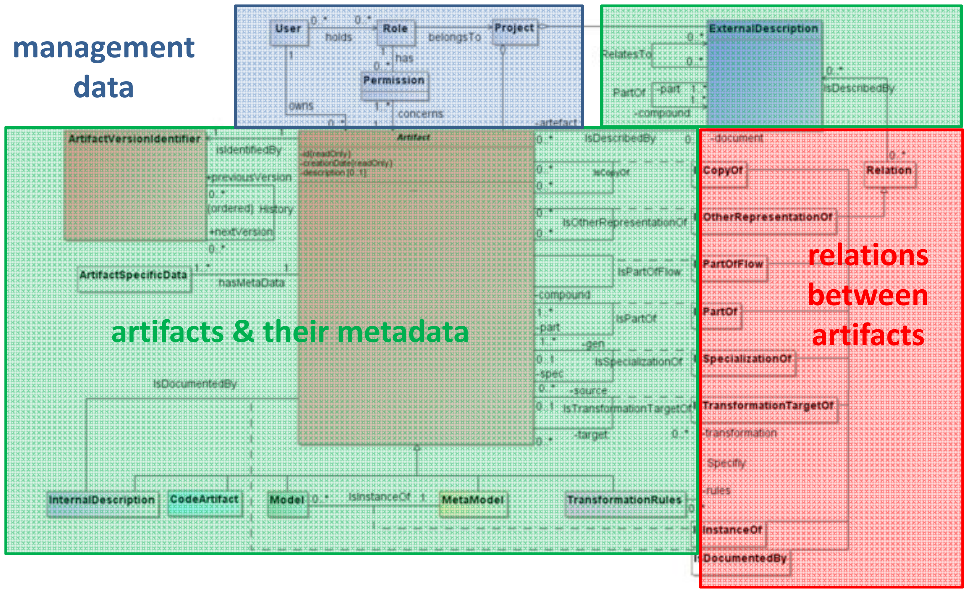
BIZYCLE

Technische Universität Berlin (TU Berlin)
Computergestützte Informationssysteme CIS / Datenbanksysteme und Informationsmanagement DIMA

Motivation & Goals

- Model-based software engineering (MBSE) consumes a large number of artifacts: (meta)models, transformation rules, code (templates), documentation, etc.
- Efficient artifact management is a significant issue in MBSE
- Goals:
  - Definition of artifacts & metadata involved in the MBSE process
  - Developing of a repository which provides support for:
    - Artifact and metadata management
    - Project/User management
    - Artifact versioning
    - Artifact (resp. version) merging
    - Consistency control
- Existing solutions (e.g., Fedora, OSCAR, ADAMS, NetBeans MDR, Adaptive Repository, IBM Rational Asset Manager, Magic Draw Teamwork Server…) do not offer these integrated capabilities

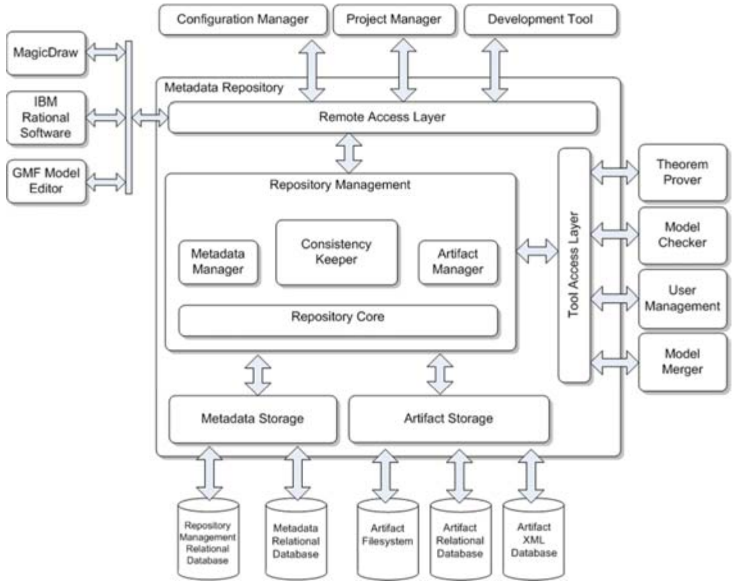
Repository Information Model



Artifact Types

- Models & Metamodels
  - UML & DSL (meta)models persisted in XMI format
- Transformation rules
  - describing model transformation steps (currently ATL)
- Code
  - incl. platform-specific deployment descriptors and templates
- Internal description
  - artifact documentation manually or automatically generated during the project
- External description
  - artifact documentation provided before the project has started
  - can be stored in external content management systems
  - e. g., requirement specification, manual, glossary, log files etc.

Repository Architecture



BIZYCLE Artifact Management

BIZYCLE Repository supports following management capabilities:

- Project Management
- User Control & Management
- Artifact Versioning (particularly Model Versioning)
- Consistency Preservation

Results achieved:

- Repository Information Model
- Repository Software Architecture
- Methodology
- Prototypical implementation

Conflict Analysis and Connector Generation within the Full BIZYCLE Integration Process

Technische Universität Berlin (TU Berlin)
Computergestützte Informationssysteme CIS / Datenbanksysteme und Informationsmanagement DIMA

Motivation

- Integrating of heterogeneous components is burdened with recurring tasks, e. g.,:
  - Understanding business semantics concerning technical interfaces
  - Combining and transforming different data types
  - Coupling conflicting functional behavior
  - Orchestrating complex call order behavior
  - Bridging incompatible communication mechanisms
- Tasks might be (semi-)automatically performed within the integration process to eliminate the integration conflicts

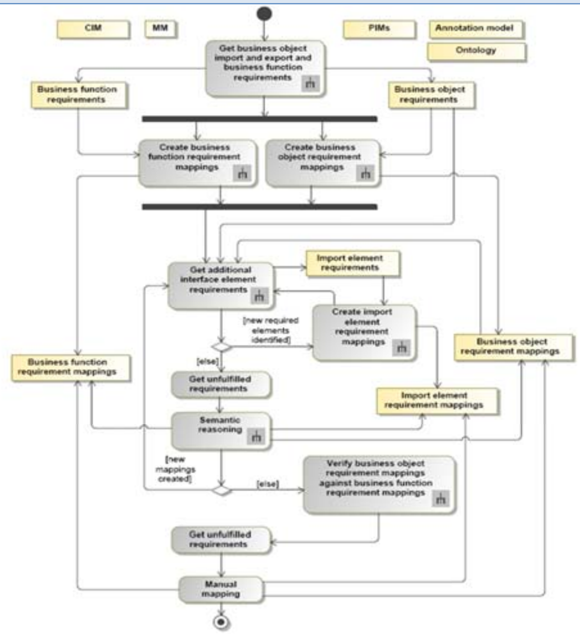
Integration Conflict Types

- Semantic Conflicts
  - Address semantical aspects of elements to be integrated, i. e., differences in the meaning they convey
- Behavior Conflicts
  - Concern dynamic aspects and are primarily caused by functional constraints of interfaces to be integrated
- Property Conflicts
  - Address the characteristics of components to be integrated, e. g., QoS properties
- Data Structure Conflicts
  - Pertain static aspects of components participating in the integration process and are caused by the differences in the data structures
- Communication Conflicts
  - Are differences in communication aspects, e. g., conflicting communication protocols

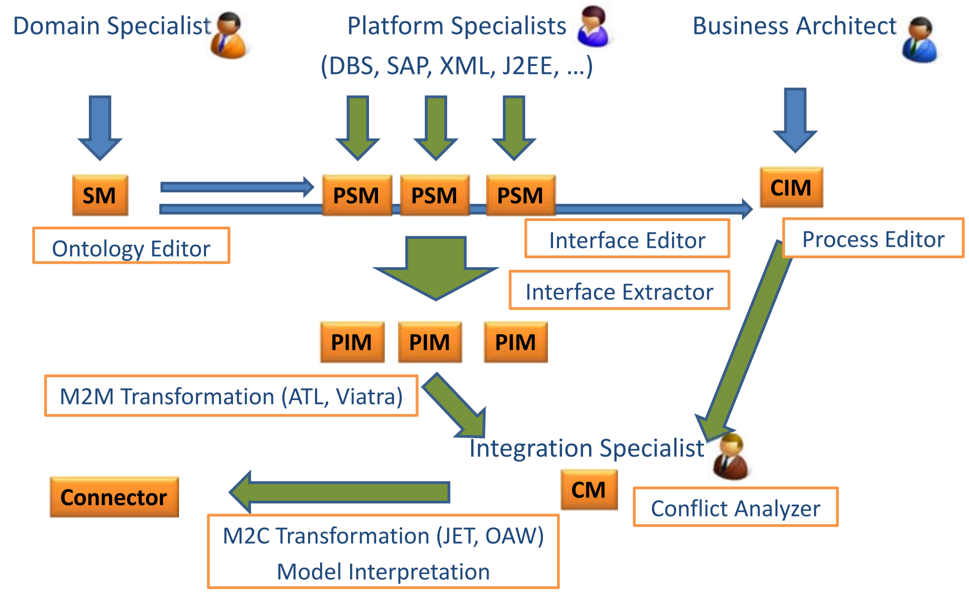
Semantic Annotations

- Data-oriented annotations are realized by references to domain objects, i.e., ontology concepts which describe data elements
  - CIM Level Annotations: Annotation of business objects, i.e., data elements participating in the integration scenario
  - PSM Level Annotations: Annotation of data elements described in various platform-specific models, e.g., data records, segments, and fields of a SAP R/3 system
  - PIM Level Annotations: Annotation of data elements described in the platform-independent models which are results of PSM-to-PIM transformation
- Function-oriented annotations are realized by references to domain functions, i.e., ontology concepts which describe actions
  - Annotation of interfaces on the PSM, PIM, and CIM levels

Semantic Conflict Analysis Algorithm

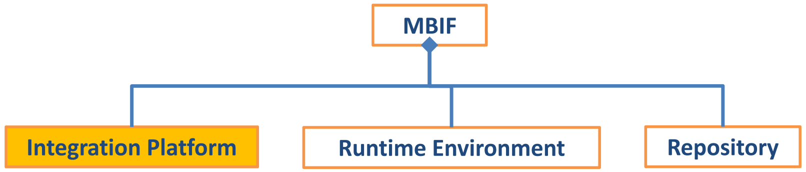


Full BIZYCLE Integration Process



Model-based Integration Framework (MBIF)

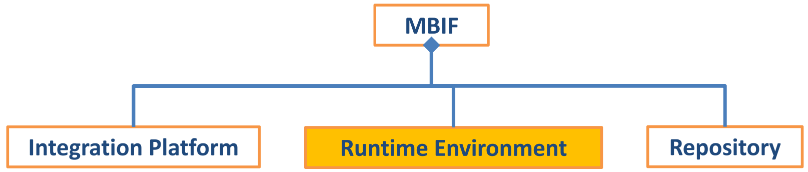
BIZYCLE



- Based on standard Eclipse
- Automatic interface extractors
- Model editors
  - Flow/semantic models (CIM, SM, part of CM): graphical (GMF)
  - Structural models (PSMs, part of CM): graphical + tree-based
- Model transformer
  - PSM + AM -> PIM + AM
  - CIM + CAM + PIMs -> CM
- Conflict analyzer/resolver
- Code generators
  - PSM -> connector code (application endpoints)
  - CM -> connector code (business logic between endpoints)

Model-based Integration Framework (MBIF)

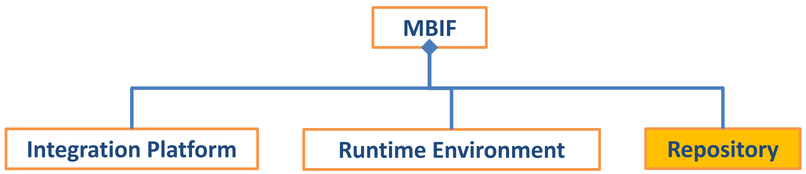
BIZYCLE



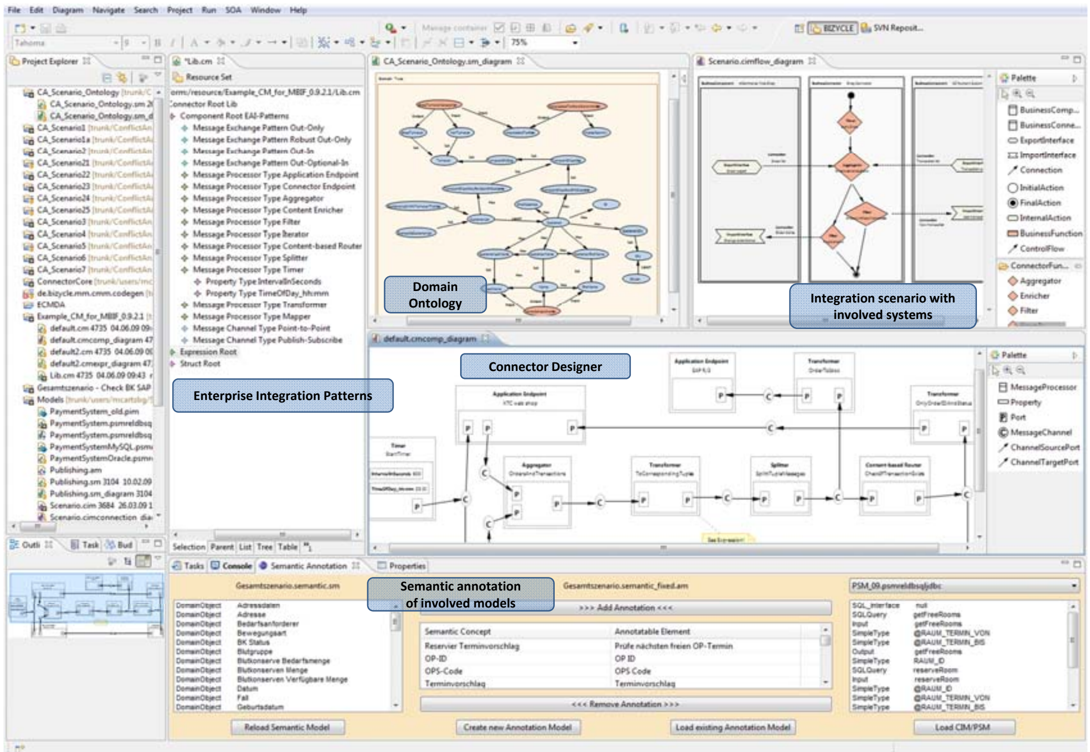
- GlassFish ESB
  - Sun-BPEL for business logic
  - Java BPEL extension for EAI patterns
  - Message- and service-oriented realization
  - Standardized target environment => portable code generators
  - Powerful QoS/management support
- Model interpreter
  - Independent Java component
  - EMF and Java Reflection for PSM interpretation
  - Low target system requirements
  - Supports online model changes

Model-based Integration Framework (MBIF)

BIZYCLE



- Subversion
  - Versioning support for all kinds of artifacts
  - Teamwork support
  - Supports artifact metadata
- Jena framework
  - Based on RDF and MySQL
  - Supports artifact relations for consistency preservation
- Apache Maven
  - Artifact relation management
  - Artifact life cycle/build management
  - Existing integration with SVN



Publications

- *Semantic Annotation and Conflict Analysis for Information System Integration*, H. Agt. G. Bauhoff,, R. Kutsche, N. Milanovic, J. Widiker, *Proc. MDTPI-Workshop @ 6th ECMDA*, Paris, France, 2010
- *Executable Domain Specific Language for Message-based System Integration*, M. Shtelma, M. Cartsburg and N. Milanovic, *ACM/IEEE 12th Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*, Denver, USA, 2009
- *Model-based Interoperability of Heterogeneous Information Systems: An Industrial Case Study*, N. Milanovic, M. Cartsburg, R. Kutsche, J. Widiker, F. Kschonsak, *Proc. 5th ECMDA*, Enschede, Netherlands, 2009
- *Model-based Semantic Conflict Analysis for Software- and Data-Integration Scenarios*, H. Agt. G. Bauhoff, J. Widiker, R. Kutsche, N. Milanovic, *Tech Report 2009-7, TU Berlin*, 2009
- *Metamodeling Foundation for Software and Data Integration*, H. Agt, G. Bauhoff, M. Cartsburg, D. Kumpe, R. Kutsche, N. Milanovic, *Proc. 8th Int. Conf. on Inf. Syst. Technology and its Applications (ISTA), at UNISCON 2009*, Sydney, Australia, 2009
- *Model & Metamodel, Metadata and Document Repository for Software and Data Integration*, N. Milanovic, R. Kutsche, T. Baum, M. Cartsburg, H. Elmasgünes, M. Pohl, J. Widiker, *Proc. ACM/IEEE 11th MODELS*, Toulouse, France, 2008
- *BIZYCLE: Model-based Interoperability Platform for Software and Data Integration*, R. Kutsche, N. Milanovic, G. Bauhoff, T. Baum, M. Cartsburg, D. Kumpe, J. Widiker, *Proc. Workshop on Model-Driven Tool- and Process Integration, 4th ECMDA*, Berlin, Germany, 2008
- *Model-based Software and Data Integration*, Kutsche R., Milanovic N. (Eds.), Conf. Proceedings MBSDI, Berlin, Germany, *Springer Verlag*, 2008
- *(Meta-) Models, Tools and Infrastructures for Business Application Integration*, Kutsche R., Milanovic N. *Proc. 7th Int. Workshop on Conceptual Modeling Approaches for e-Business* (*EComo 2008*), Klagenfurt, Austria, 2008

AGENDA of the eBISS Tutorial on MBSDI

- History & Background
- Goal: Model Based Software & Data Integration
- Languages and Models
- Metalanguages and Metamodels
- BIZYCLE: Model Based Software & Data Integration
- **BIZWARE: Domain Specific Languages**

BIZWARE @ TUB 2010 - 2013

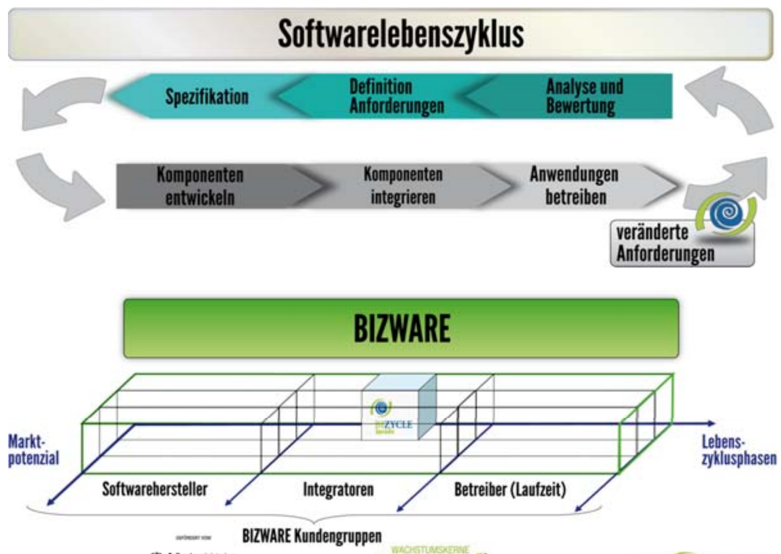
BIZWARE

Domain Specific Languages
"Methods, Languages, Tools and Infrastructure for the Model and Software Factory of the BIZWARE Regional Project Group"

Our Project team @ TU Berlin:

Dr. Ralf-D. Kutsche, Dr. Nicole Natho
Henning Agt, Yan Li (partially), Yuexiao Li
+ many part-time students:
Amauri Albuquerque, Andreas Büscher, Nico Franzeck, Amir Matallaoui, Silvia Sandy-Martinez, Kamran Mohtadi, Lakshmi Vuyyuru, Andreas Wolf

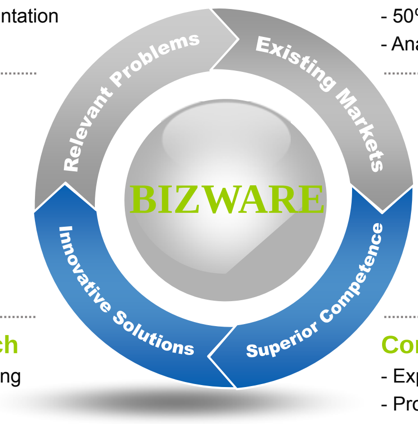
BIZWARE
Solution for the Complete Software Life Cycle



BIZWARE Market Relevance

Solve Real Problems
- Expensive software life cycle
- Inadequate business orientation
- Incompatible devices

Relevant Markets
- AIM market 9.4 bn. €, COTS 7.4 bn. €
- 50% market opportunity for innovators
- Analysts forecast prospects

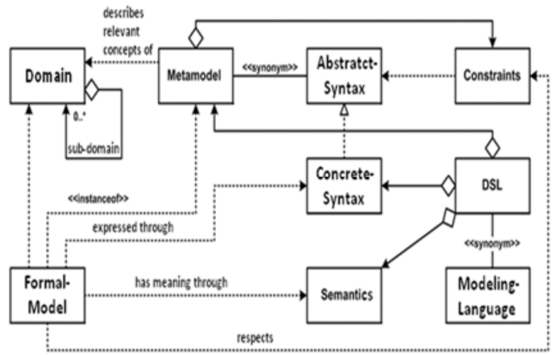


Innovative Approach
- Consequent meta modeling
- Software generation
- Tailor-made tools

Competence
- Experienced BIZYCLE Consortium
- Proficient new partners
- Broad access to branches

What are domain specific modeling languages?

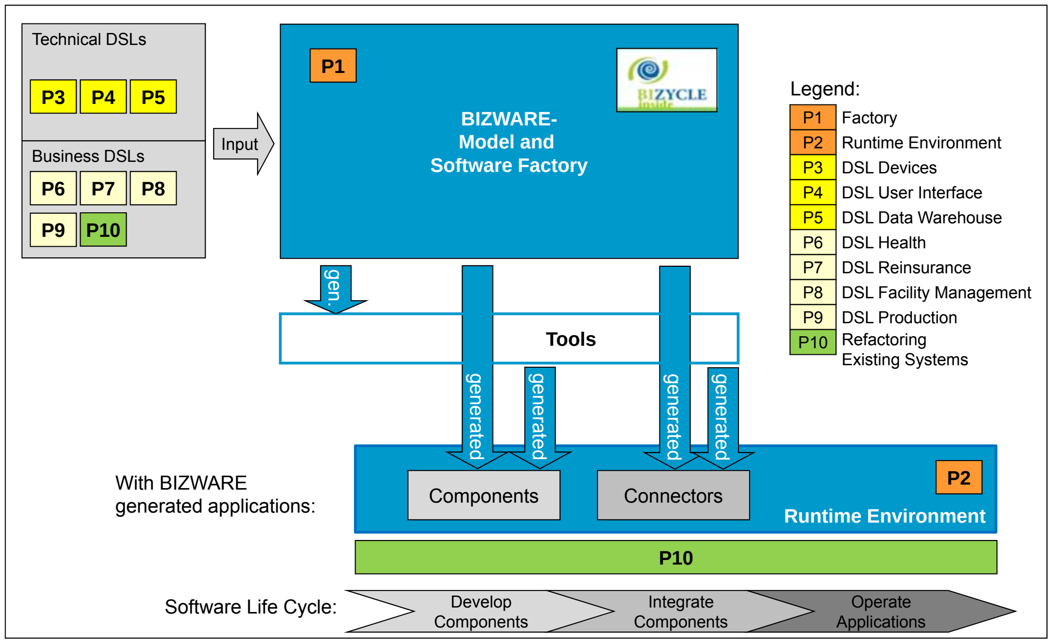
- Small modeling languages, tailored for a specific domain
- Domain specific (graphical) notation
- Easier modeling for domain (NOT modeling!) experts
- Reuse, Combination, …
- Defining an DSML:
  - Abstract syntax
  - Concrete syntax
  - Language constraints (Semantics)



General Purpose Mod. Languages (like UML) vs. DSLs

UML and DSL: the Lego Metaphor

- UML is like a pile of basic LEGO blocks with few colors, sizes and shapes
- DSL is like the special LEGO kit for Medieval Knight Castle
- With the basic LEGO kit you can build only average castles
- With the special Castle kit you can build excellent castles
- On the other hand, with the basic LEGO kit you can also build average cars or planes
- But with the special Castle kit you can build only really terrible cars or planes



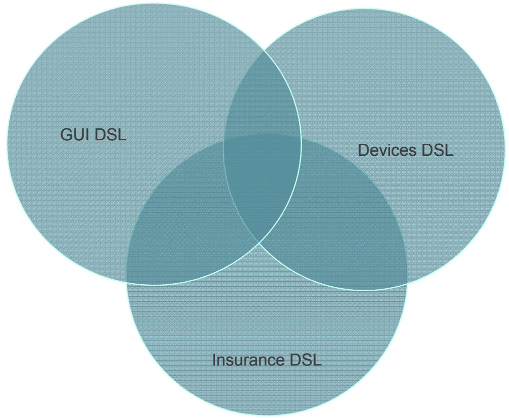
BIZWARE

Protopype of a DSL for GUIs
- Graphical User Interface Development based on Wireframes –

Collaboration TU Berlin and Akquinet/Tech@Spree

BIZWARE GUI DSL & Other DSLs

… finally to be integrated with insurance applications and mobile devices…



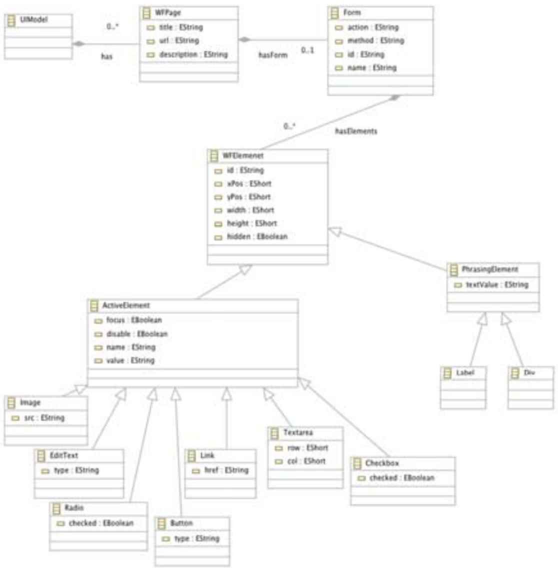
GUIs in Different Devices

Classical computers, tablets, smart phones, technical controllers, healthcare devices, etc. …

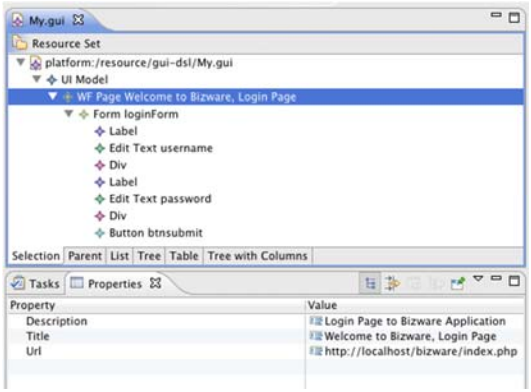
Basic GUI Elements

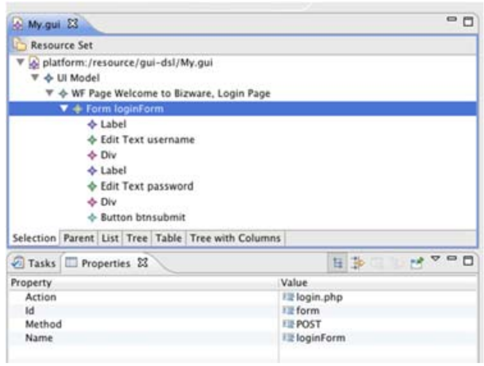
- Shapes, Colours, Frames, …
- Text Fields
- Images
- Tables
- Combo Boxes
- Buttons
- ...

GUI Meta Model



Abstract Syntax emf





Abstract Syntax Xtext

Our Application

```
UIModel:
    'uimodel' name = ID
    '{'
        wfpage += WFpage*
    '}'
;
```

Wireframe Page

```
WFpage:
    'wfpage' name = ID
    '{'
        'title' title=STRING';'
        form += Form*
    '}'
;
```

```
Form:
    'Form' (name = ID)?
    'method' method =STRING
    'action' action =STRING
    ('class' style = STRING)?
    ('id' id = ID)?
    '{'
        (wfelements += WFElements";")*
    '}'
;
```

```
WFElements:
        Button | Input | Radio | Checkbox| TextArea | SelectBox | DataEntry | Img | Link | YNQuestion | Div | Label
;
```


Abstract Syntax Xtext

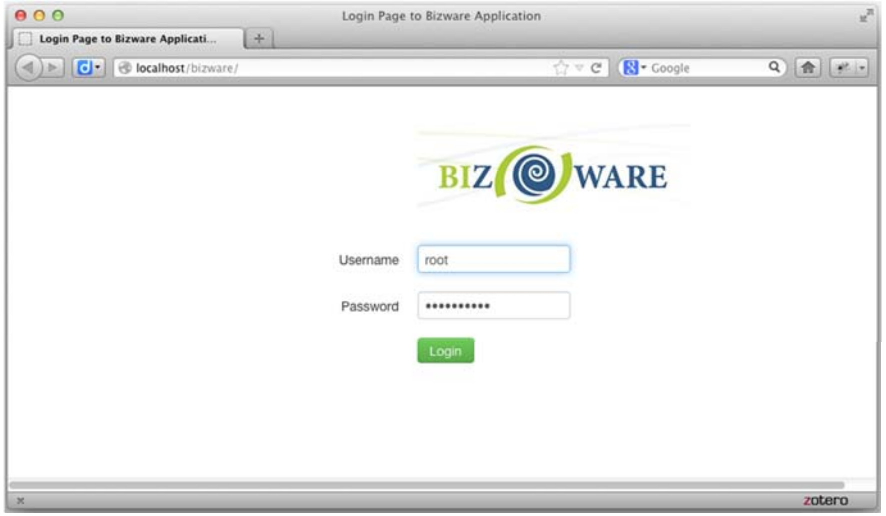
```
DataEntry:
    'dataEntry' (name = ID)?
    ('id' id = ID)?
    ('text' text = STRING)?
    type = (TextArea | SelectBox | Input)
;
```

Enter Your name:

```
dataEntry enterName text "Enter Your name" input required Text;
```

<div class="row" class="control-group">
<div class="span4" class="control-label">Enter Your name:</div>
<div class="span5"><input class="input-large" name="I;enterName" id="I;" type="text" required placeholder""></div>
</div>

Example - Login Page



BIZWARE

Case study: Domain specific language in facility management (DSLFM)

Master thesis Kamran Mohtadi

Definition: Facility Management

- "Facility management is a profession that encompasses multiple disciplines to ensure functionality of the ***built environment*** by integrating people, place, process and technology." (International facility management association)
- Dimensions of facility management according to European Network of Facility management / Standard EN15221: (http://www.eurofm.org/knowledge/en15221/)
  - Space and infrastructure
  - People and organization

in detail:

- **Cleaning services**
- **Pest control**
- **Catering & vending services**
- **Furniture procurement**
- **Office supplies**
- **Maintenance**
- **Logistics**
- **Event management**
- **Energy management**
- **Utilities**
- **Landscaping & parking**
- **Design and construction services**
- **Lease**
- **Rental & space management**
- **Corporate real estate services**
- **Workplace management**
- **Safety & security services**
- **(internal) Relocation services**
- **Document management**
- **Hospitality/reception management**
- **Travel services**
- **Property administration**

Computer Aided Facility Management (CAFM)

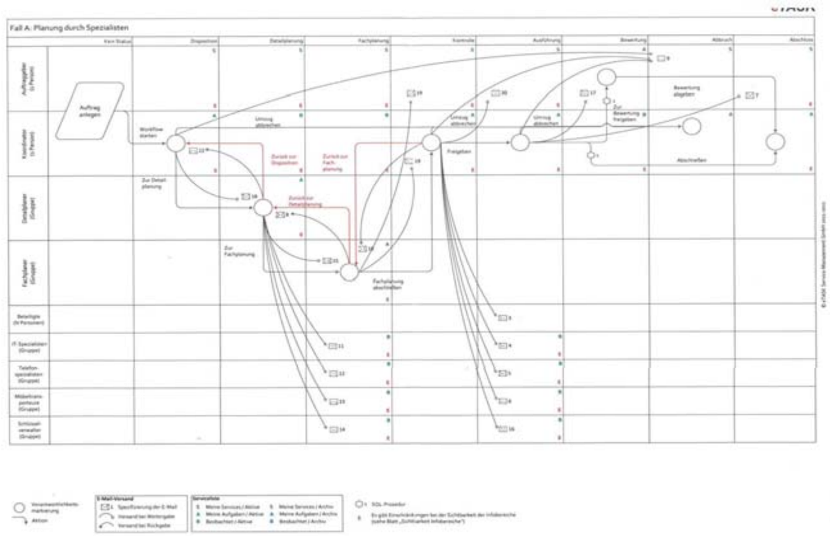
- CAFM (Computer Aided Facility Management Software) or TIFM (Total Integrated Facilities Management Software) software supports the specific processes of facility management and also the people involved in these processes
- Core processes supported in CAFM systems:
  - Stock documentation, land management, move management, contract management, rent management, operating cost management, cleaning management, key management, energy controlling and maintenance management.
- eTask.FM-Portal: central platform structure as the basis for the connection of all other modules.
- eTask.FM-Portal is based on several data tables, which store data regarding processes, people, emails, activities, …

Case study: DSLs in Facility Management (DSLFM)

- Status before introducing DSLFM:
  - **Central database is configured manually for each new process**
  - **Manual configuration according to MS Visio based process / data flow diagrams called "Infographs"**
  - **Infograph models widely accepted within the company and by the customers**
- Problems:
  - **Manual configuration of the database for each scenario change is time consuming and error-prone**
  - **Identification of similar processes**
- Goals of the master thesis:
  - **Automation of database configuration**
  - **Keeping the graphical notation of Infograph models**
  - **"Real" modeling environment including code generation (MS Visual Studio)**

Case study: DSLFM – Infograph models

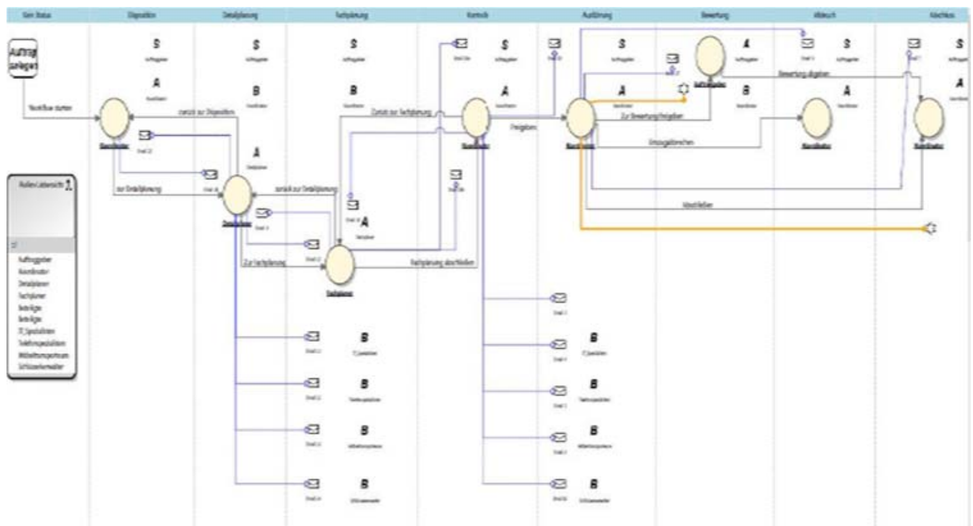
- Domain analysis through Infograph models and additional data from eTask



Results: DSLs in Facility Management (DSLFM)

- DSLFM / Model transformation from Infographs to "real" FM Models in MS Visual Studio
  - A Domain Specific Language for Facility Management
  - Developed in MS Visual Studio DSL Tools
  - Similar GUI to Infograph models
  - Proved to automatically generate in the biggest given scenario up to 11000 lines of SQL code for database configuration
  - Avoidance of errors through constraints and validity checks
  - Already in productive use by eTask in a few scenarios, replacing infograph models and manual configuration of the eTask.FM-Portal DB

Results: DSLs in Facility Management (DSLFM)



BIZWARE

Automated Construction of a Large Semantic Network of Related Terms for Domain-Specific Modeling

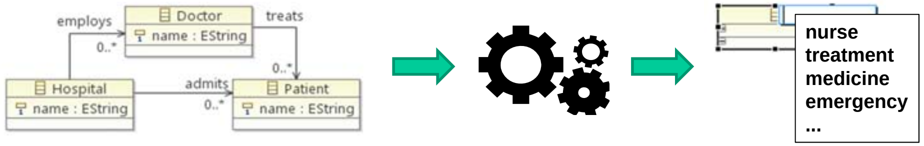
@ CAiSE 2013, Valencia, Springer LNCS 7908

Forthcoming Ph.D. thesis Henning Agt

Motivation

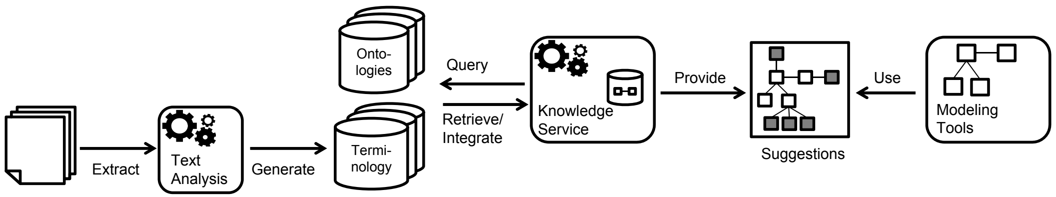
- Autocompletion applications

- Predict what the user wants next in completion of a model

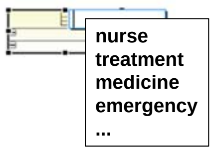


Research Goals

- **Vision 1: Provide automated suggestions of semantically related model elements for domain modeling** (focus on domain terminology and conceptual design)
- **Vision 2: Try to remove errors from domain models** (focus on relationships)

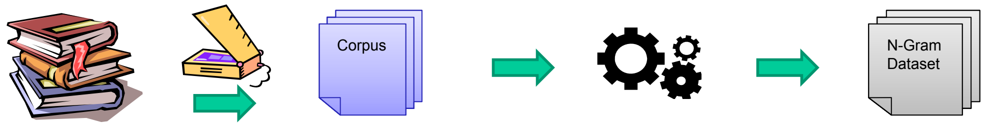


- Requirements for the intended application
  - Dictionary of terms
  - Relations between terms
  - Query interface and ranking functions



Google Books N-Gram Dataset

- Large amounts of text data



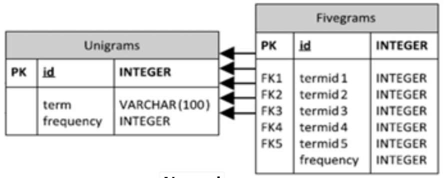
- N-Grams
  - Sequence of *n* consecutive words/tokens and its frequency
  - Google provides 1,2,3,4 and 5-grams in several languages
- Using the English-All dataset V2 (1-grams and 5-grams)

...	
to go to the hospital	46,410
general condition of the patient	28,198
I was in the hospital	19,268
discharge from the hospital .	12,476
admission to the hospital .	10,558
the patient to the hospital	6,422
by placing the patient in	6,026
between doctor and patient .	5,908
...	

...	
able to leave the hospital	4,629
patient admitted to the hospital	4,303
a patient in the hospital	3,844
the symptom of the patient	2,559
the patient under local anesthesia	2,536
a patient is suffering from	2,475
the doctor and the hospital	1,362
the hospital and the doctor	1,017
...	

SemNet Preprocessing

- N-gram database
  - → Make the data manageable
  - *Input*: 2.5 terabytes of text
  - *Output*: Tables with 10 million 1-grams and 710 million 5-grams (21 gigabytes)



- Part-of-speech tagging [8], [9]
  - → Identify lexical category of each text token
  - *Output*: Table with POS tags for each 5-gram (14 gigabytes)

Adjective · Normal Noun · Preposition · Determiner

JJ NN IN DT NN
general condition of the patient
NN NN NN CC NN
drug store pharmacist or doctor

Coordinating conjunction

- Normalization
  - → Reduce amount of word variations
  - Plural stemming, lowercasing of adjectives and normal nouns
  - Proper nouns are not touched

doctors	→ doctor
Medical practitioner	→ medical practitioner
hospitals in Valencia	→ hospital in Valencia

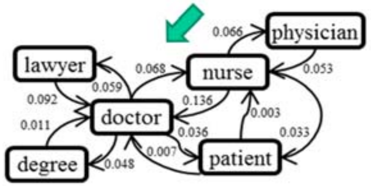
- Result: 710 million normalized and tagged 5-grams

SemNet Construction

- Discard 5-grams that contain 4 or 5 stopwords
  - to go to the doctor · I am what I am · a) (2)
- Apply pattern matching on the remaining 5-grams
  - → Result: Large table of binary relations
- Frequency aggregation
  - Many terms co-occurred in different contexts
- Relative frequency computation
  - For each term with respect to its related terms
- Graph construction
  - Directed, weighted edges
  - Relational database and graph database serialization (SQLite / Neo4J)

term	rel. term	frequency
doctor	nurse	18,656
doctor	nurse	18,022
doctor	degree	16,094
doctor	lawyer	13,258
nurse	doctor	13,258
nurse	doctor	4,135
nurse	patient	9,750
nurse	physician	12,355

term	rel. term	frequency
doctor	nurse	783,395
doctor	lawyer	685,529
doctor	degree	555,031
nurse	doctor	783,395
nurse	physician	383,167
nurse	patient	188,288



Querying SemNet

- Query Interfaces
  - SQL: Query the relational database
  - Cypher: Query the Neo4J database
  - Java: Use SemNet in your applications
  - PHP: Explore the data in a web interface

```
select * from nouncooccurrences where termw1 = 5824331 and termw2 is null and termw3 is null order by relfreq desc limit 20;
```

```
Public ArrayList<String>
getRelatedStringTerms(ArrayList<String> inputTerms) { … }
```

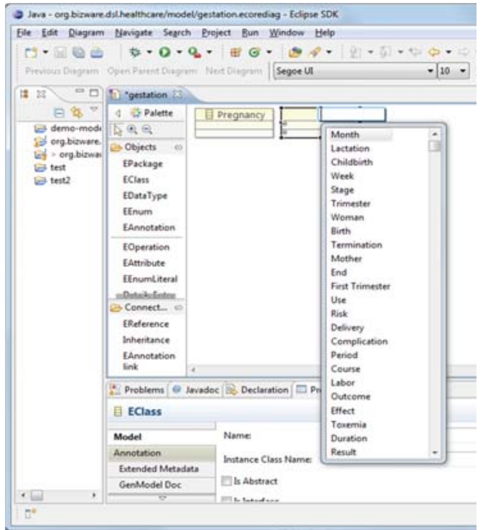
- Examples of top 10 automatically identified related terms

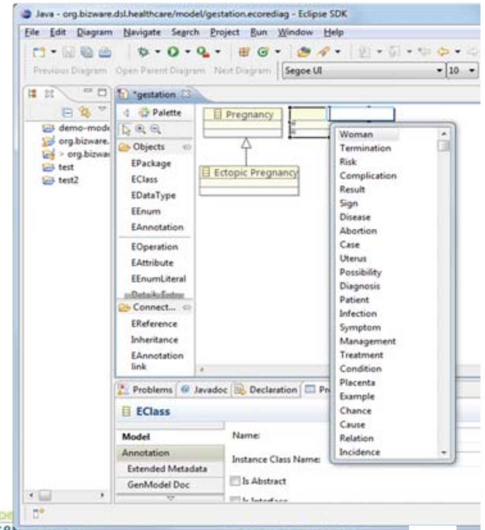
	teacher	doctor	electricity	software engineering	lymphocytic choriomeningitis
f	32.4M	19.1M	7.2M	212K	23K
1	student	nurse	water	CASE	virus
2	parent	lawyer	gas	field	LCM
3	school	degree	quantity	component	mouse
4	pupil	office	magnetism	area	cell
5	child	patient	heat	computer science	syngeneic
6	administrator	hospital	use	component	cytotoxicity
7	role	teacher	conductor	system	mediated cytotoxicity
8	training	order	current	discipline	mumps
9	work	law	steam	aspect	lymphocyte
10	principal	dentist	amount	term	monkey
$\#r$	8728	5519	2716	144	31

(f – absolute term frequency in the original text corpus, #r – number of related terms)

Modeling With Semantic Autocompletion

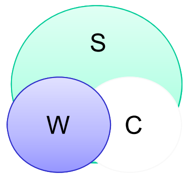
- Prototype: Ecore Diagram Editor with class name suggestions
- Automated suggestion adaption with respect to the content of the model

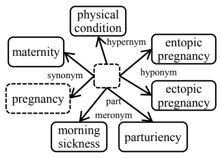




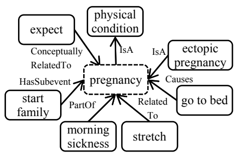
Evaluation Setup

- Challenge
  - No gold standard available for many information extraction tasks
- Our strategy: Compare SemNet to existing knowledge bases
  - Provide measurements on how much information of WordNet and ConceptNet is contained in SemNet
- WordNet V3.0: Lexical database for the English language [16]
  - Synsets: Grouped words that share the same sense
  - Relations: Mainly taxonomic, part-whole and synonyms
- ConceptNet V5.1: Semantic graph for general human knowledge [17]
  - Nodes: Any natural language phrase that expresses a concept
  - Relations: Taxonomic, part-whole, related-to and several others
- SemNet: Semantic Network of Related Terms
  - Nodes: Noun terminology
  - Relations: Probabilistic links

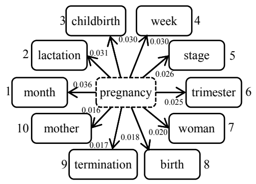




Word sense pregnancy in WordNet
(7 out of 32 relations)



Concept pregnancy in ConceptNet
(7 out of 58 relations).



Term pregnancy in SemNet
(First 10 out of 4039 relations).

Summary BIZWARE Strategy

- Continuation of the experiences in BIZYCLE on a new basis:
  - Addressing the complete software development cycle, but (feasible) restrictions of manageable domains: analysis, requirements definition, specification, design, development, integration, operation, evaluation, etc.
- Focus on the foundation of domain-specific languages:
  - Meta modeling, model instantiation, model transformation, multi modeling, model integration, semantic modeling (ontologies, semantic annotations, etc.), test generation, process support

BIZWARE – Research results

Diploma and Master Theses 2011 - 2013

- **Michael Stollhans** (Diploma thesis, TU Berlin, Oct. 2011, Advisor: Dr. Ralf-D. Kutsche) "Modellierungsmethodik für fachliches, domänenspezifisches Monitoring "
- **Benedict Wandelt** (Diploma thesis, TU Berlin, June 2012, Advisor: Henning Agt) „Metamodelle und Ontologien für domänenspezifische Sprachen“ .
- **Amir Matallaoui** (Master thesis, TU Berlin, Feb. 2013, Advisor: Dr. Nicole Natho) „Domain-Specific Languages: Development Guidelines & Tool Support Comparison“ .
- **Lakshmi Vuyyuru** (Master thesis, Univ. Potsdam/TU Berlin, March 2013, Advisors: Dr. Nicole Natho, Dr. Ralf-D. Kutsche/TUB, collab. Prof. Tiziana Margaria/Univ. Potsdam) „Re-establishing Vertical Consistency Between Meta-Models and their Corresponding Models“ .
- **Kamran Mohtadi** (Diploma thesis, TU Berlin, April 2013, Advisor: Dr. Ralf-D. Kutsche) „Development of Domain Specific Languages in the Software Industry - a Practical Case Study”.
- **Silvia Teresa Sandy Martinez** (Master thesis, TU Berlin, 2013, in progress, Advisor: Henning Agt, Dr. Ralf-D. Kutsche) “Patterns in Domain Models”. To be finished in July 2013

BIZWARE – Research Results

Scientific Conferences 2011 - 2013

- H. Agt: Supporting Software Language Engineering by Automated Domain Knowledge Acquisition. MODELS 2011 Workshops Wellington, New Zealand, Oct. 16-21, 2011. Proc. LNCS 7167, Springer 2011
- H. Agt, R. Kutsche, T. Wegeler: Guidance for Domain Specific Modeling in Small and Medium Enterprises. SPLASH '11 Workshops; Portland, USA, Oct. 22 -27, 2011.
- H. Agt, R. Kutsche, N. Natho, Y. Li: The BIZWARE Research Project. In: Model Driven Engineering Languages and Systems MODELS 2012, 15th Int. Conf. - Exhibition Track, Innsbruck , Austria, Sept. 30 - Oct. 05, 2012.
- H. Agt: SemAcom: A System for Modeling with Semantic Autocompletion. In: Model Driven Engineering Languages and Systems MODELS 2012, 15th Int. Conf. - Demo Track, Innsbruck, Austria, Sept. 30 - Oct. 05, 2012.
- H. Agt, R. Kutsche: Automated Construction of a Large Semantic Network of Related Terms for Domain-Specific Modeling. In: Advanced Information Systems Engineering CAiSE 2013, 25th Int. Conf., Valencia, Spain, June 17-21, 2013.