Modellalapú szoftvertervezés

Tematika, követelmények

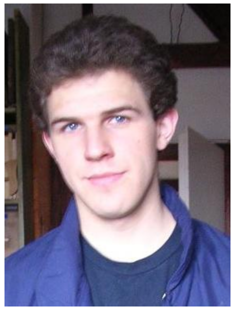
Varró Dániel
varro@mit.bme.hu

A tárgyról

- Modellalapú szoftvertervezés =
  Model-driven Software Design (MDSD)
- Előadások
  - MDSD: Kedd 10:15-12:00, IB413
  - (SzolgInt: Hétfő 10:15-12:00, IB413)
- Gyakorlatok
  - Közös: Csütörtök 14:15-16:00, IB413
  - Előadások+gyakorlatok pontos beosztása a honlapon, Google Calendarban
- Honlap
  - http://www.inf.mit.bme.hu/edu/courses/mdsd

A tárgy oktatói

- Előadók
  - Varró Dániel
  - Horváth Ákos
  - Ráth István
  - + meghívott előadók
- Gyakorlatok:
  - Hegedüs Ábel
  - Izsó Benedek
  - Ujhelyi Zoltán
  - Semeráth Oszkár
  - Szárnyas Gábor



Követelmények

- Mottó: „a gyakorlat teszi a mestert”
  - A félév során 3 összefüggő házi feladatot kell megoldani
    - Határidőre
    - 2 fős csapatmunkában
  - Alapelv
    - modellezés + validálás + kódgenerálás
    - a megvalósítás minőségének javítása céljából
  - Kiírás: https://www.inf.mit.bme.hu/edu/courses/mdsd/homework (hamarosan)
- Szóbeli vizsga, a házi feladatok eredménye beszámít
  - Extra feladatok a félév során a megajánlott jegyért
- Segédanyagok: Fóliák +
  - Angol könyv: Model driven software engineering in practice by Brambilla, Cabot and Wimmer (fóliák is részben)

Ütemterv

- Modellezés (6. hét: március 18.):
  - Követelmények
  - Domain modellezés (metamodellezés)
  - Állapotgépek
- Domain-specifikus nyelvek (12. hét, április 29.)
  - Kódgenerálás
  - Modellvalidáció
- Implementáció (14. hét, május 17.)
  - Integrált rendszer bemutatása

Gyakori kérdések

- Hogyan alakítsunk csapatot?
  - Rátok bízzuk. Két fős csapatok legyenek.
  - A csapat nevét és a tagok névsorát feb. 13-ig küldjétek el emailben a ahorvath@mit.bme.hu címre. (jobb lenne már ma)
- Mi lesz, ha valaki nem talál magának csapatot?
  - Majd mi találunk neki. (De csak végső esetben.)
- Mikor lesz az első gyakorlat?
  - Február 14: Bevezető, SVN+Trac használata.
- Hogyan kell beadni a házi feladatot?
  - A határidő előtti nap éjfélig SVN-be feltölteni,
  - A beadás külön időpontban lesz.
  - Mindketten legyetek jelen.
  - Ügyeljetek a precíz munkanapló vezetésére!
    (Részletek az első gyakorlaton)

Tanácsok

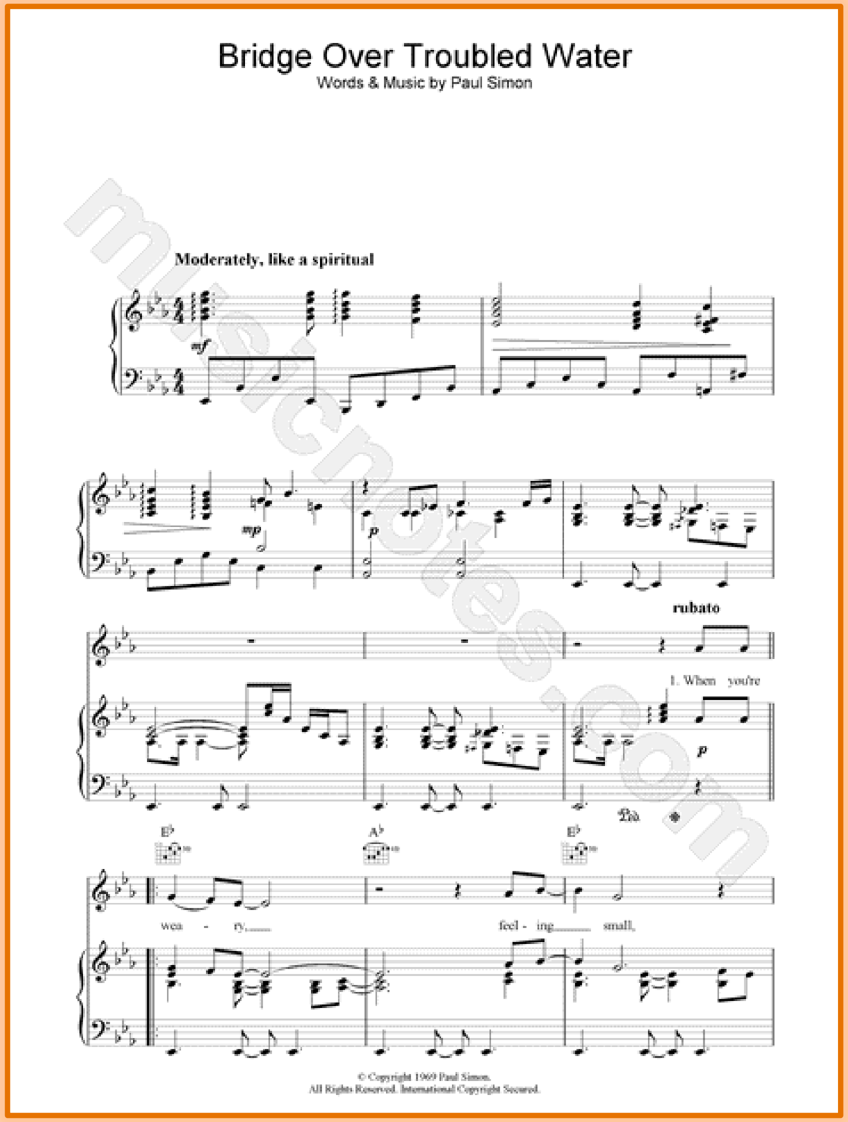
- A legfontosabb: a házi feladatot időben el kell kezdeni!
- Használjátok ki a lehetőségeket:
  - A gyakorlatokon bemutatjuk a technológiákat.
  - A demonstrátoroktól nyugodtan lehet emailben segítséget, tanácsokat kérni.
- Vegyétek komolyan a csapatmunkát!
  - Ellenőrizni fogjuk, és figyelembe vesszük a végső értékelésnél.
- Olvassátok el figyelmesen a feladatkiírásokat a honlapon!
  - A házi feladat nem pusztán szakmai kihívás,
  - projekttervezési és munkaszervezési is!
- Az előadásokra határozottan megéri bejárni.
  A vendégelőadókéra különösen!
- Mindig kérdezzetek bátran: modeling@sauron.inf.mit.bme.hu

MDSD tematika

- Bevezető, a modellvezérelt fejlesztés alapjai
- Szoftvermodellezés
  - Követelmény analízis
  - Domain modellezés (metamodellezés)
  - Jólformáltsági kényszerek: OCL
  - Dinamikus modellezés (állapottérképek)
  - Architektúra modellezés (kritikus rendszerekben)
- Domain-specifikus nyelvek
  - Az Eclipse Modeling Framework
  - Automatikus kódgenerálási technikák
  - Modell-lekérdezések (EMF-IncQuery keretrendszer)
  - Modelltranszformációk gráftranszformációk által
- Szoftverfejlesztési módszertanok és a modell-alapú megközelítés (SPEM, DO-331)
- Modell-menedzsement
- Meghívott előadás

Song writing methods of Simon and Garfunkel

Paul Simon’s technique #1


Bridge Over Troubled Water
Words & Music by Paul Simon
musicnotes.com
Moderately, like a spiritual
mf
mp
p
rubato
1. When you're
E♭
A♭
wea - ry,
feel - ing small,
© Copyright 1969 Paul Simon.
All Rights Reserved. International Copyright Secured.

1. Create music first

Bridge Over Troubled Water

When you're weary
Feeling small
When tears are in your eyes
I will dry them all

I'm on your side
When times get rough
And friends just can't be found
Like a bridge over troubled water
I will lay me down
Like a bridge over troubled water
I will lay me down

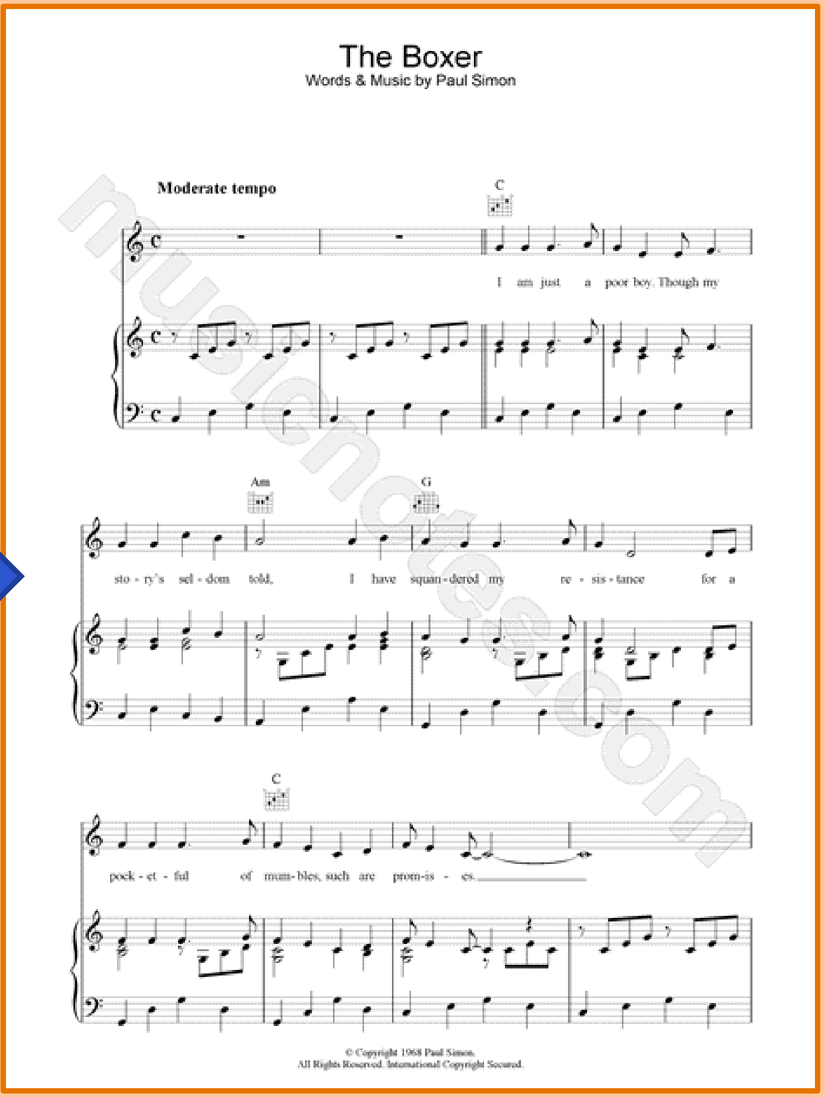
2. Write lyrics accordingly

Paul Simon's technique #2

The Boxer

I am just a poor boy
Though my story's seldom told
I have squandered my resistance
For a pocket full of mumbles such are promises
All lies and jests
Still a man hears what he wants to hear
And disregards the rest
When I left my home and my family
I was no more than a boy
In the company of strangers
In the quiet of the railway station running scared
Laying low, seeking out the poorer quarters
Where the ragged people go
Looking for the places only they would know

1. Write lyrics first



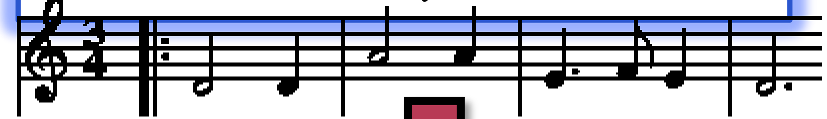
The Boxer
Words & Music by Paul Simon
Moderate tempo
C
I am just a poor boy. Though my
Am
G
sto - ry's sel - dom told, I have squan - dered my re - sis - tance for a
C
pock - et - ful of mum - bles, such are prom - is - es.
© Copyright 1968 Paul Simon.
All Rights Reserved. International Copyright Secured.
musicnotes.com

2. Compose music accordingly

A Combined Technique…

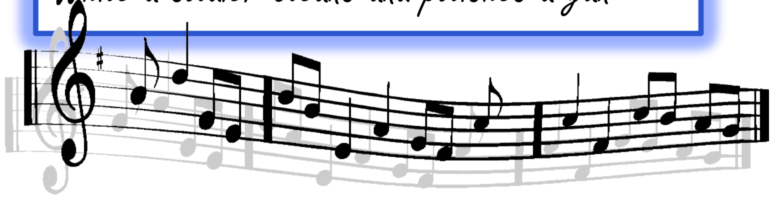
Scarborough Fair (Folk Song)

Tell her to find me an acre of land,
Parsley, sage, rosemary and thyme;
Between the salt water and the sea strand,
Then she'll be a true love of mine.



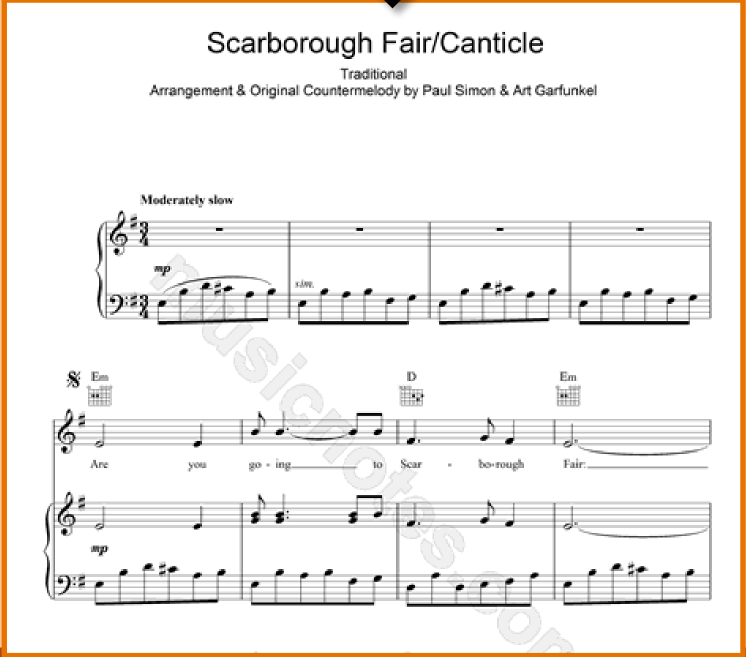
The Side of a Hill (P. Simon)

On the side of a hill, a little cloud weeps
And waters the grave with its silent tears
While a soldier cleans and polishes a gun



Canticle (rearranged by A. Garfunkel)

On the side of a hill a sprinkling of leaves
Washes the grave with silvery tears
A soldier cleans and polishes a gun



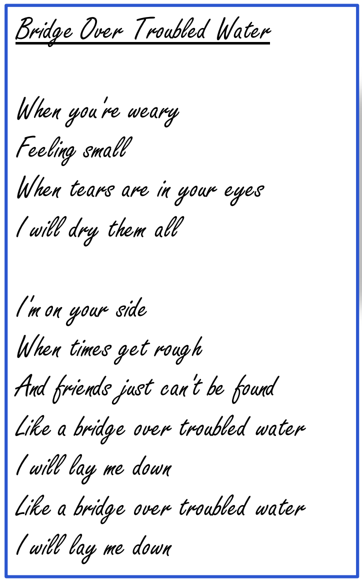
Scarborough Fair/Canticle

Traditional
Arrangement & Original Countermelody by Paul Simon & Art Garfunkel

Naming These Techniques…

Bridge Over Troubled Water
Words & Music by Paul Simon
Moderately, like a spiritual
rubato
1. When you're
wea - ry,
feel - ing small,

1. Create music first



Bridge Over Troubled Water

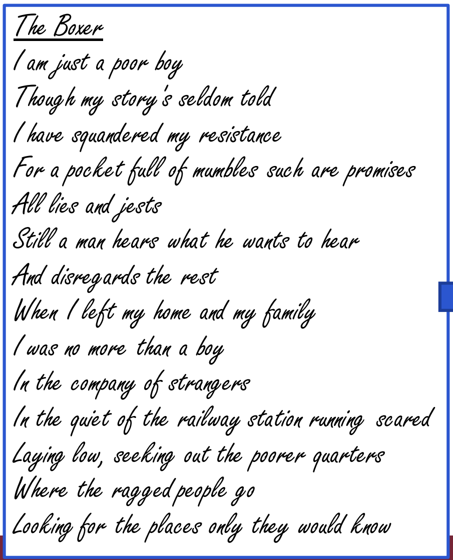
When you're weary
Feeling small
When tears are in your eyes
I will dry them all

I'm on your side
When times get rough
And friends just can't be found
Like a bridge over troubled water
I will lay me down
Like a bridge over troubled water
I will lay me down

2. Write lyrics accordingly

Music Driven Song Development (MDSD)

Lyrics Driven Song Development



The Boxer
I am just a poor boy
Though my story's seldom told
I have squandered my resistance
For a pocket full of mumbles such are promises
All lies and jests
Still a man hears what he wants to hear
And disregards the rest
When I left my home and my family
I was no more than a boy
In the company of strangers
In the quiet of the railway station running scared
Laying low, seeking out the poorer quarters
Where the ragged people go
Looking for the places only they would know

1. Write lyrics first

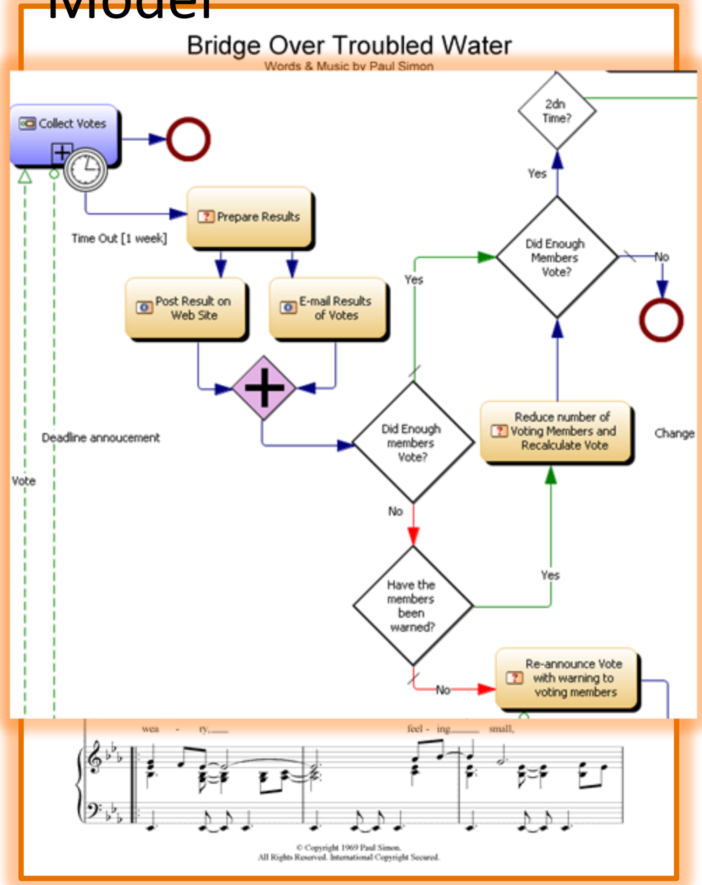
The Boxer
Words & Music by Paul Simon
Moderate tempo
I am just a poor boy. Though my
sto - ry's sel - dom told,
I have squan - dered my re - sis - tance for a
pock - et - ful of mum - bles, such are prom - is - es

2. Compose music accordingly

Applying the Principle to Software Systems

MDSD = Model Driven Software Development / Engineering

Model


Bridge Over Troubled Water
Words & Music by Paul Simon
Collect Votes
Prepare Results
Time Out [1 week]
Post Result on Web Site
E-mail Results of Votes
Deadline annoucement
Vote
2dn Time?
Yes
Did Enough Members Vote?
No
Yes
Did Enough members Vote?
Reduce number of Voting Members and Recalculate Vote
Change
No
Have the members been warned?
Yes
No
Re-announce Vote with warning to voting members
© Copyright 1969 Paul Simon. All Rights Reserved. International Copyright Secured.

Code

Bridge Over Troubled Water

```
import com.lauchenauer.istockhelper.
import com.lauchenauer.lib.ui.Vertic
import com.lauchenauer.lib.util.Brow
public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;
  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);
    initUI();
  }
  protected void initUI() {
    setSize(440, 600);
    Container cont = getContent
    JPanel p =
```

Like a bridge over troubled water
I will lay me down

Music

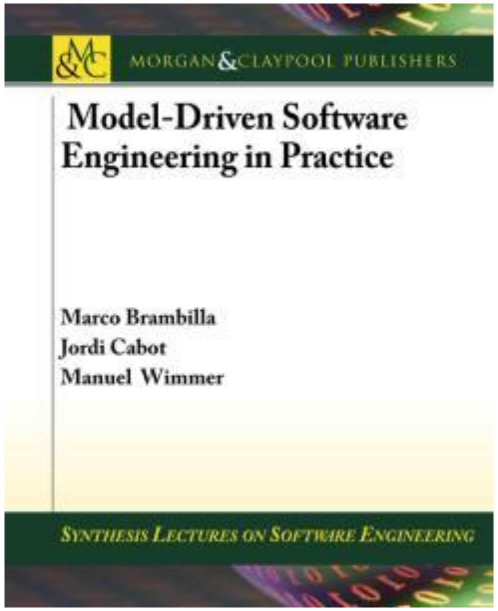
Lyrics

MDSD principles

MODEL-DRIVEN SOFTWARE ENGINEERING IN PRACTICE

Marco Brambilla,
Jordi Cabot,
Manuel Wimmer.
Morgan & Claypool, USA, 2012.

www.mdse-book.com
www.morganclaypool.com
or buy it on www.amazon.com


MORGAN & CLAYPOOL PUBLISHERS

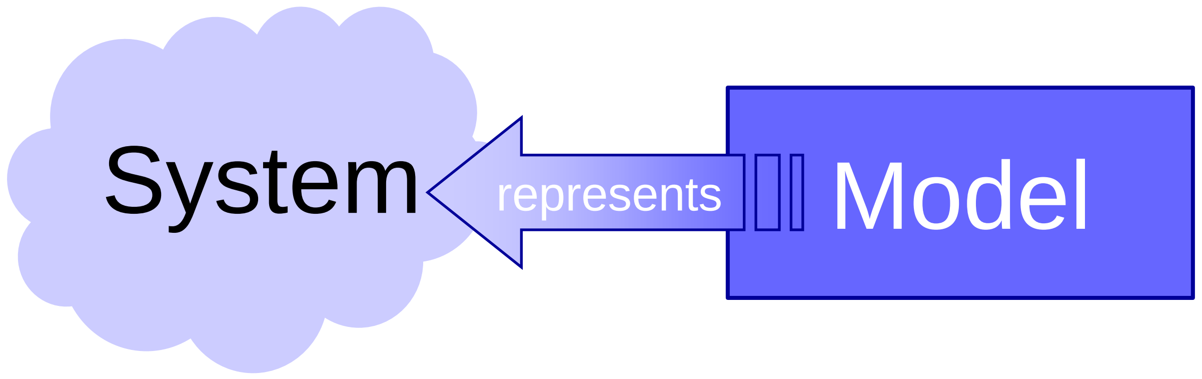
Model-Driven Software Engineering in Practice

Marco Brambilla
Jordi Cabot
Manuel Wimmer

SYNTHESIS LECTURES ON SOFTWARE ENGINEERING

Models

What is a model?



Mapping Feature	A model is based on an original (=system)
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 an original with respect to some purpose

Purposes:

- descriptive purposes
- prescriptive purposes

MDSE Equation

Models + Transformations = Software

Modeling Languages

- **Domain-Specific Languages (DSLs):** languages that are designed specifically for a certain domain or context
- DSLs have been largely used in computer science. Examples: HTML, Logo, VHDL, Mathematica, SQL

- **General Purpose Modeling Languages** (GPMLs, GMLs, or GPLs): languages that can be applied to any sector or domain for (software) modeling purposes
- The typical examples are: UML, Petri-nets, or state machines

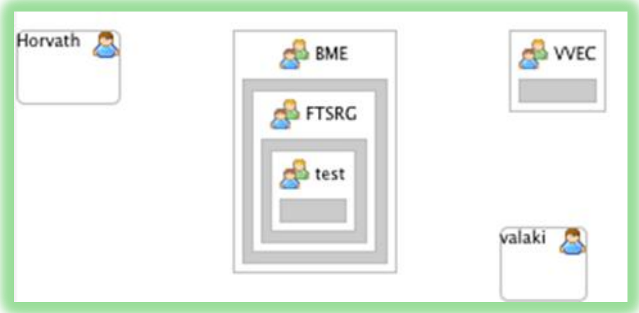
Types of models

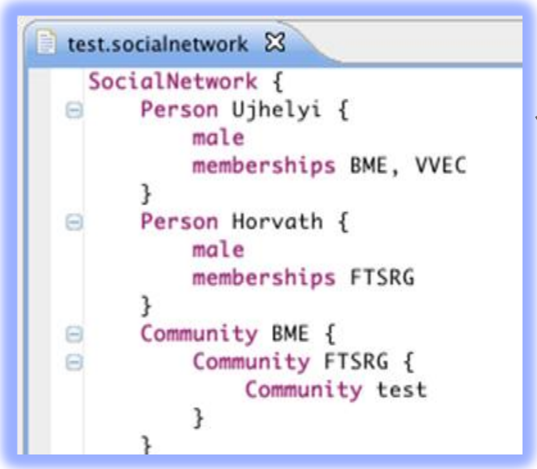
- **Static models:** Focus on the static aspects of the system in terms of managed data and of structural shape and architecture of the system.
- **Dynamic models:** Emphasize the dynamic behavior of the system by showing the execution

- Just think about UML!

Domain Specific Modeling Languages

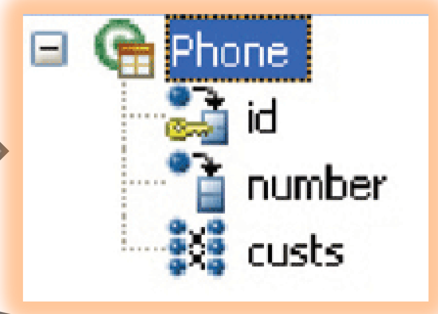
Concrete syntax
(Graphical/Textual)



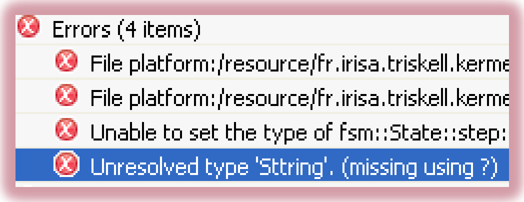


```
test.socialnetwork
SocialNetwork {
    Person Ujhelyi {
        male
        memberships BME, VVEC
    }
    Person Horvath {
        male
        memberships FTSRG
    }
    Community BME {
        Community FTSRG {
            Community test
        }
    }
```

Abstract syntax
(Metamodel)



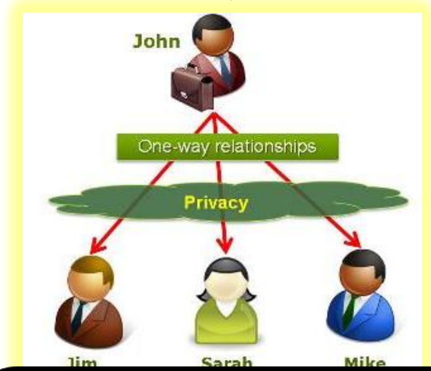
Well-formedness
constraints



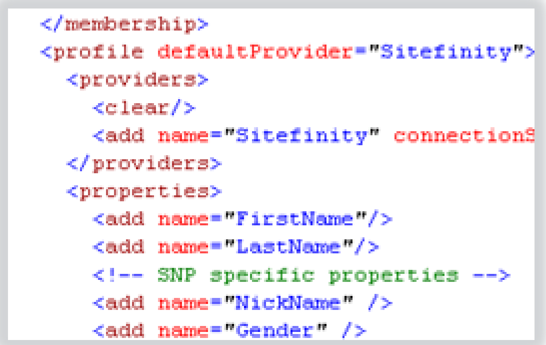
```
Errors (4 items)
    File platform:/resource/fr.irisa.triskell.kerme
    File platform:/resource/fr.irisa.triskell.kerme
    Unable to set the type of fsm::State::step:
    Unresolved type 'Sttring'. (missing using ?)
```

Behavioural semantics,
Simulation

Mapping



Code
generation



```
</membership>
<profile defaultProvider="Sitefinity">
  <providers>
    <clear/>
    <add name="Sitefinity" connectionS
  </providers>
  <properties>
    <add name="FirstName"/>
    <add name="LastName"/>
    <!-- SNP specific properties -->
    <add name="NickName" />
    <add name="Gender" />
```

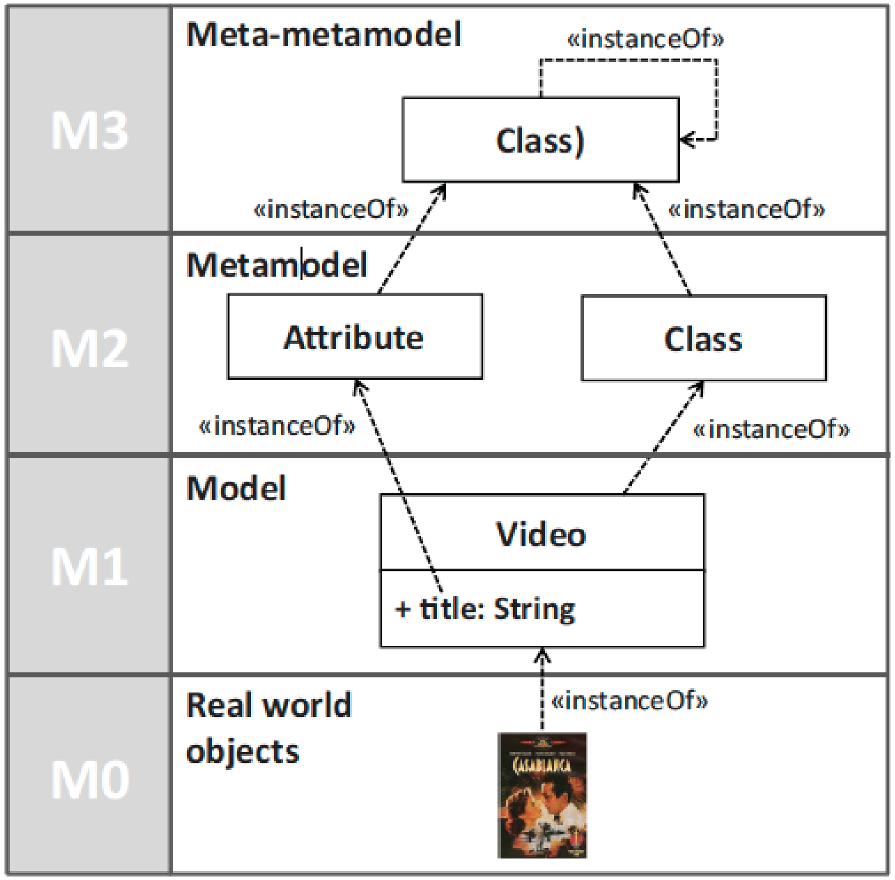
Foundations of many modern tool
(design, analysis, V&V)

- Domains: avionics, automotive, business modeling

Metamodeling

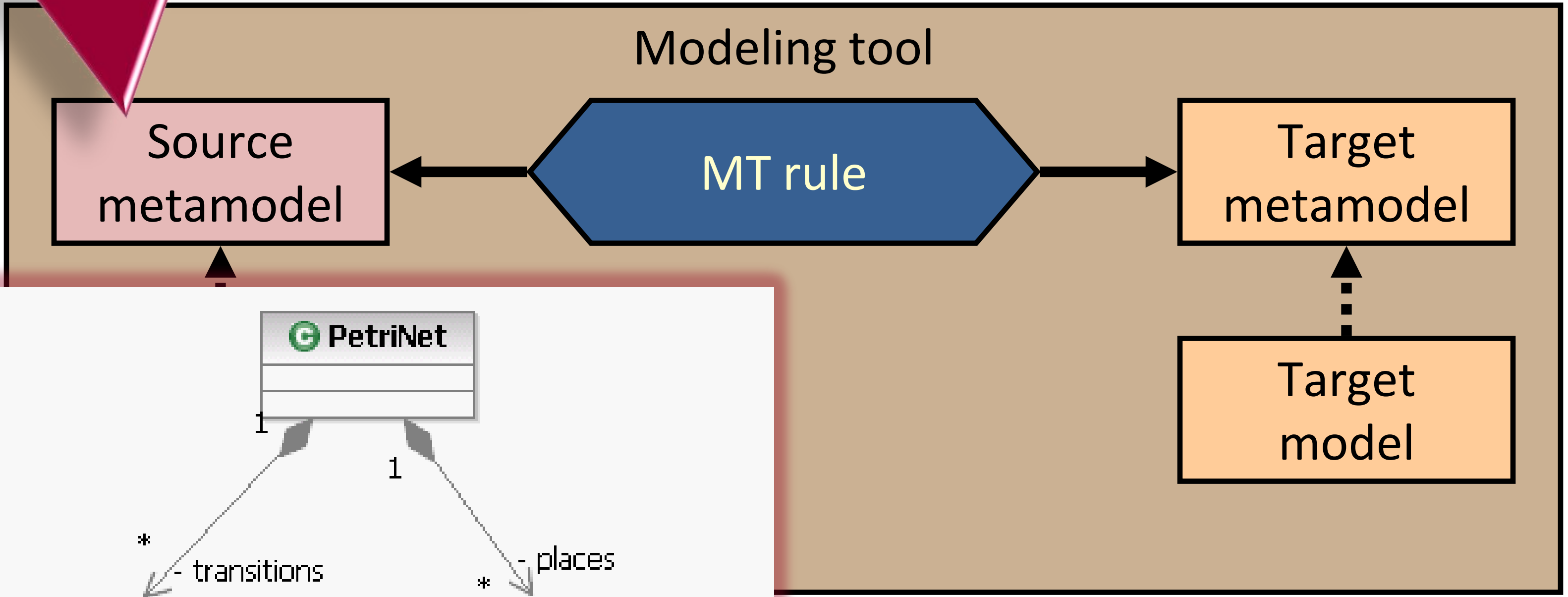
- To represent the models themselves as "instances" of some more abstract models.
- **Metamodel** = yet another abstraction, highlighting properties of the model itself

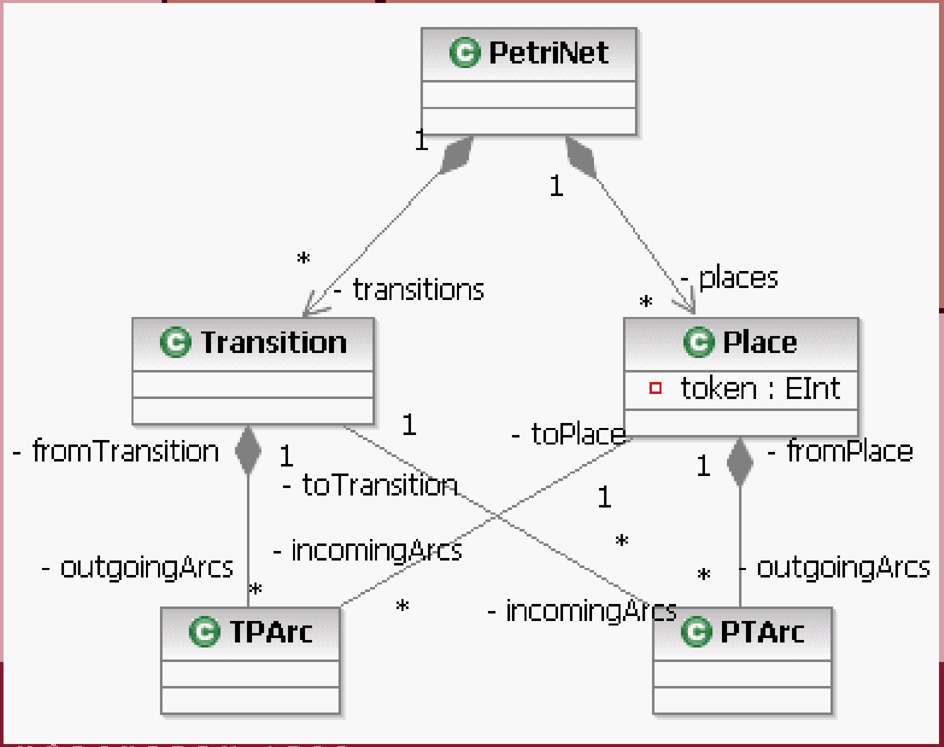
- Metamodels can be used for:
  - defining new languages
  - defining new properties or features of existing information (metadata)



Model Transformation Overview: Metamodels

Metamodel: Precise spec of a modeling language

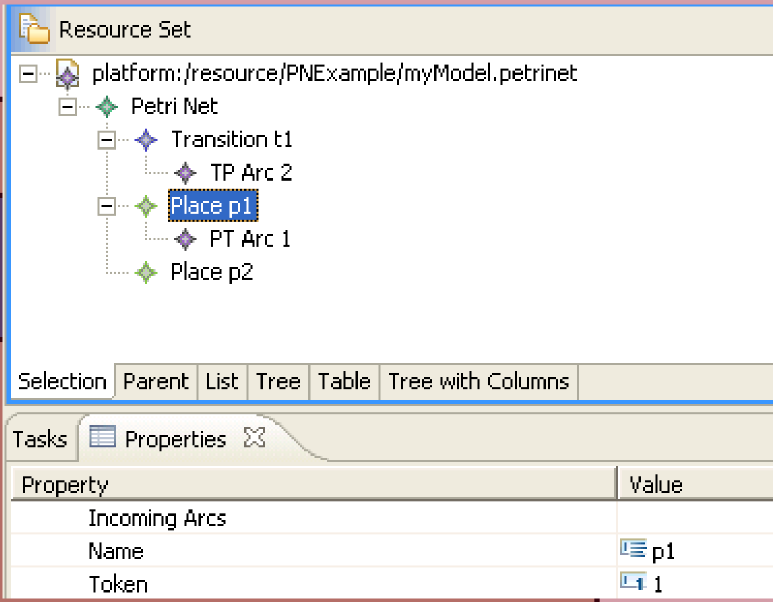




Model Transformation Overview: Models

Eclipse Modeling Framework (EMF):

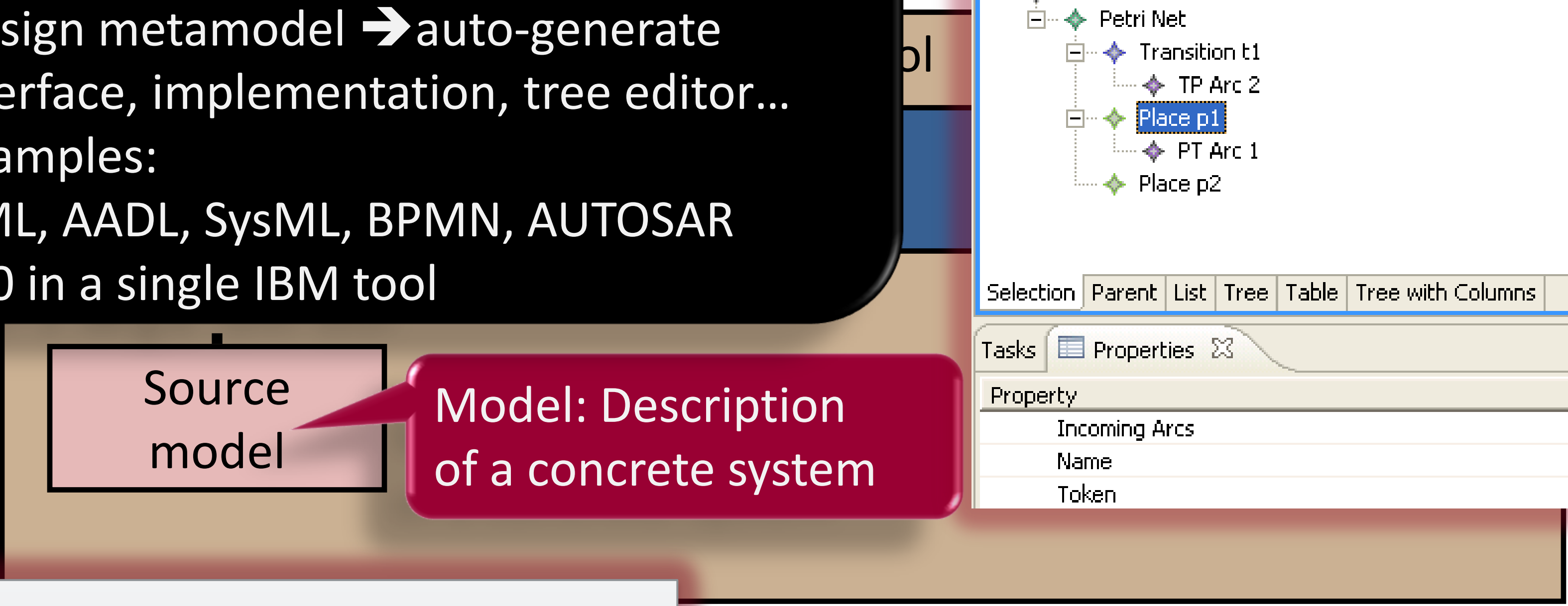
- De facto modeling standard for Eclipse based modeling tools
- Design metamodel ➔auto-generate interface, implementation, tree editor…
- Examples:
  UML, AADL, SysML, BPMN, AUTOSAR
  >30 in a single IBM tool


Resource Set
platform:/resource/PNExample/myModel.petrinet
Petri Net
Transition t1
TP Arc 2
Place p1
PT Arc 1
Place p2

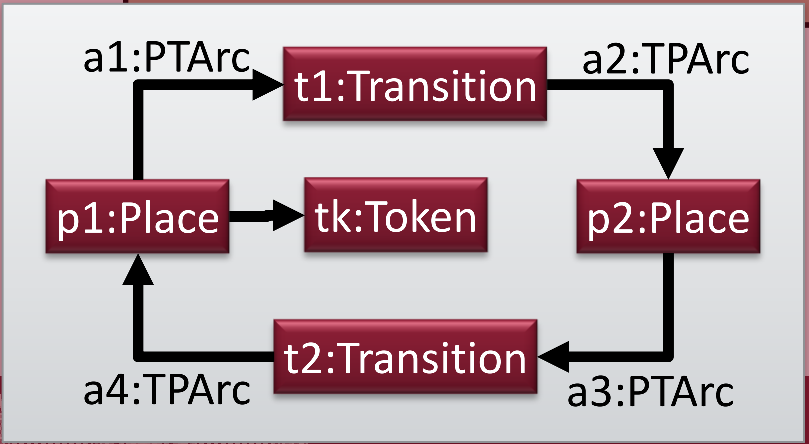
Selection | Parent | List | Tree | Table | Tree with Columns

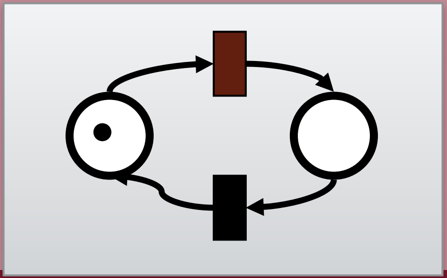
Tasks | Properties

Property	Value
Incoming Arcs	
Name	p1
Token	1



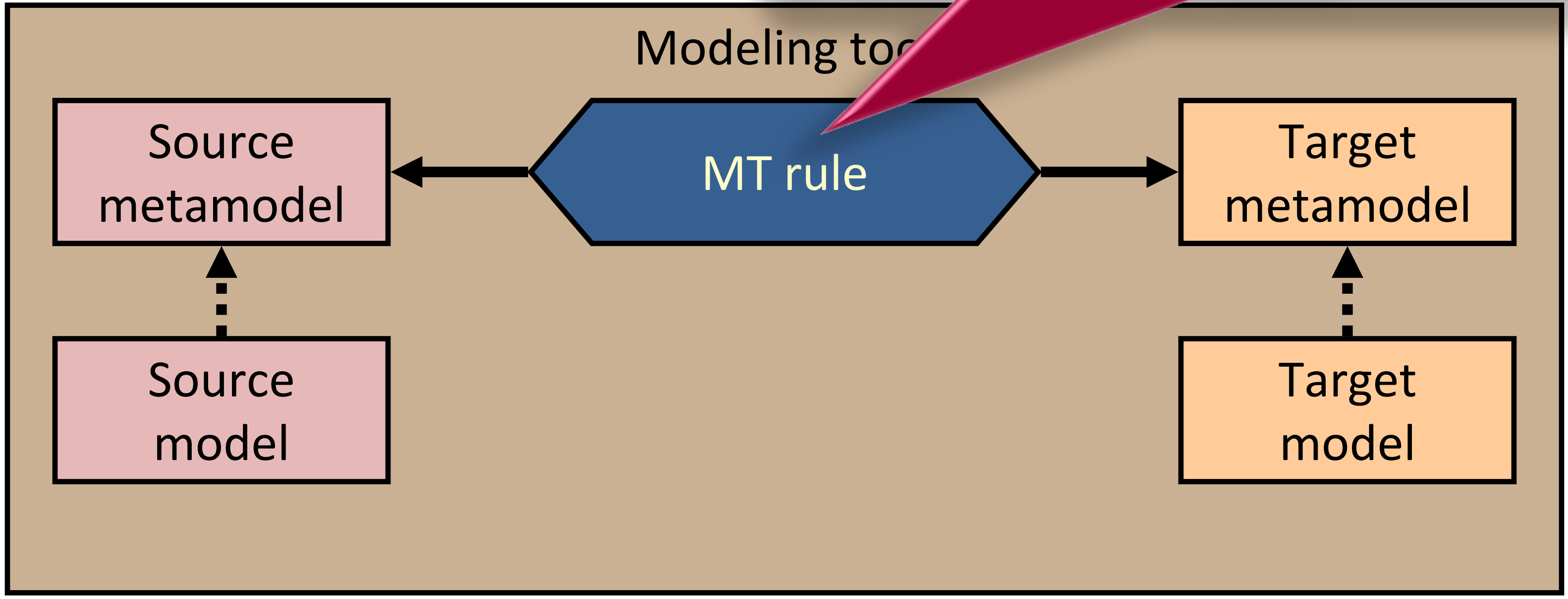
Model: Description of a concrete system

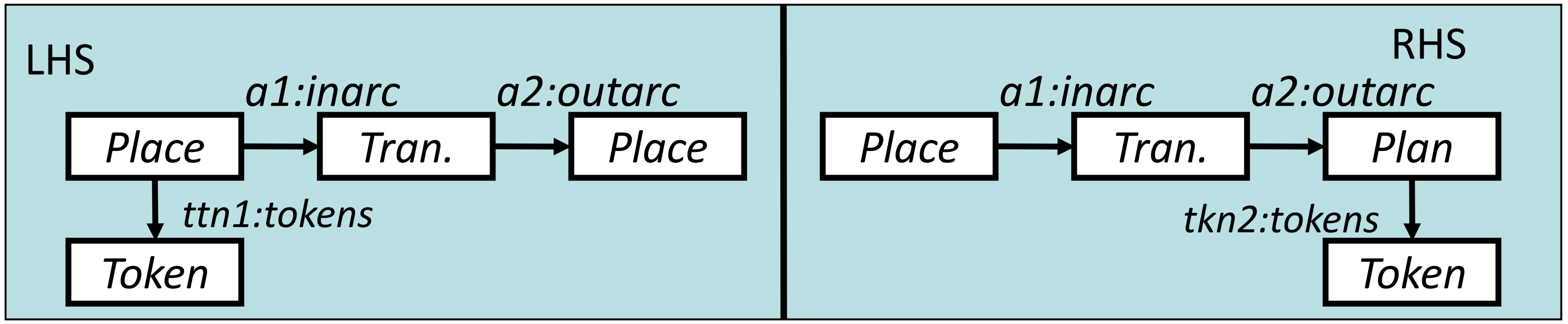




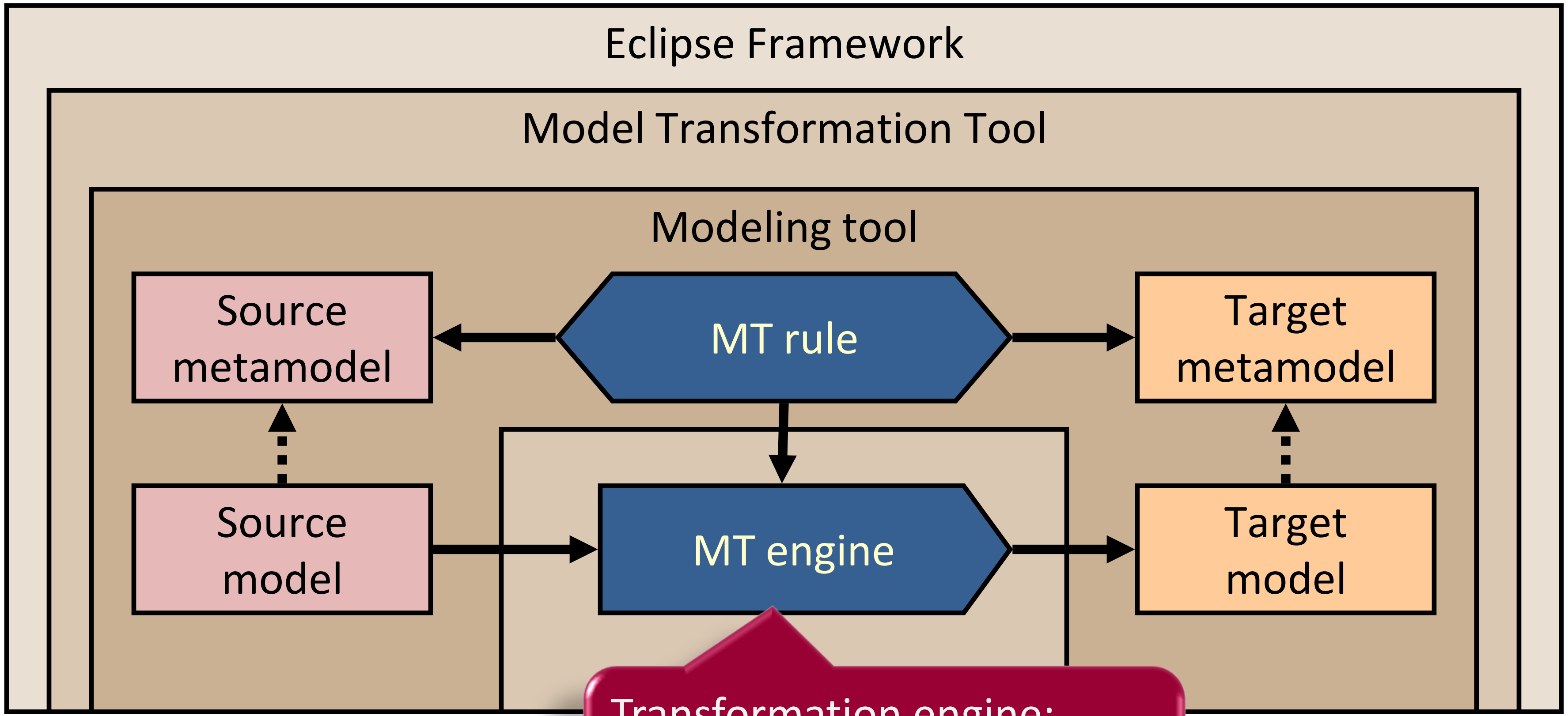
Model Transformation Overview: Rules

Model Transformation:
How to generate a target equivalent of an arbitrary source model





Model Transformation Overview: Rule Execution



Transformation engine: Support for querying and manipulating large models

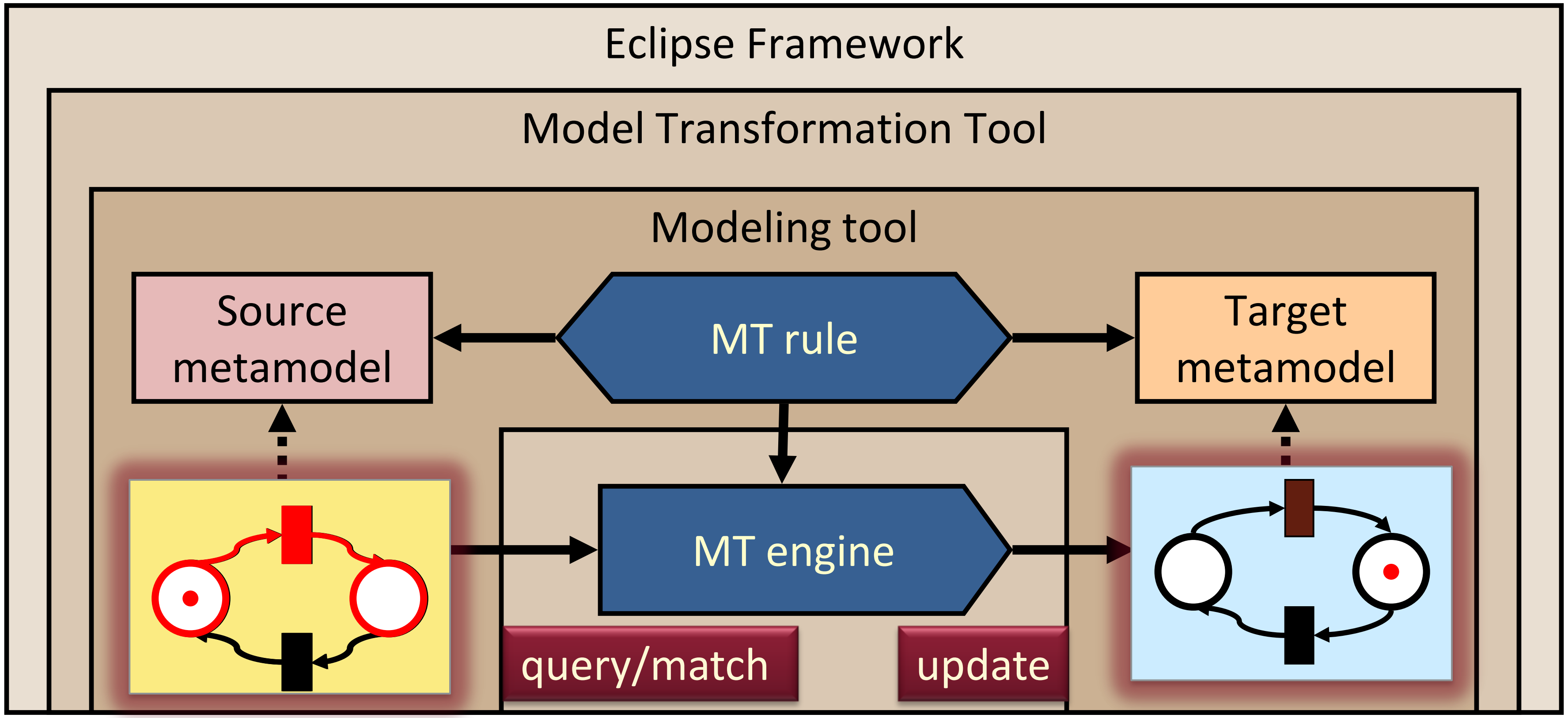
EMF-IncQuery:
http://www.eclipse.org/incquery/

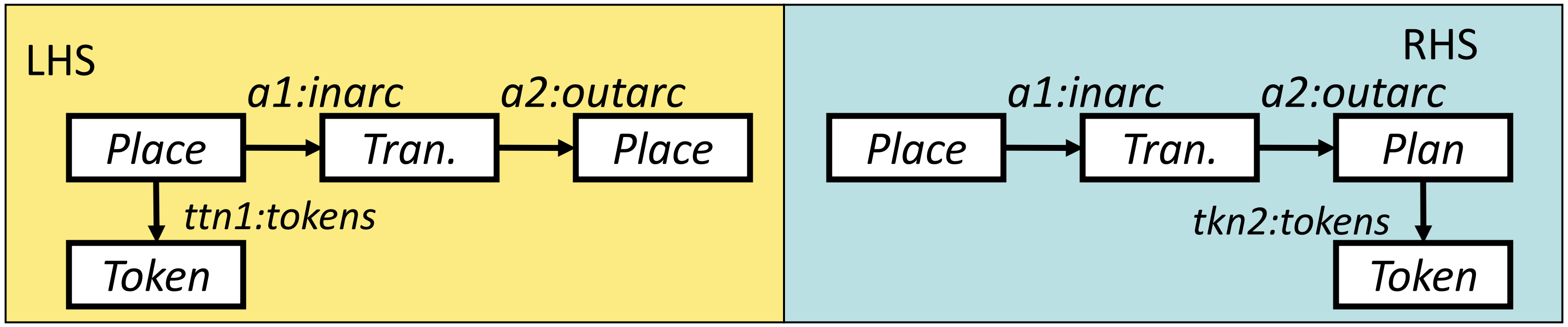
VIATRA2:
http://www.eclipse.org/gmt/VIATRA2/

ATL

QVT

Model Transformation by Graph Transfromation



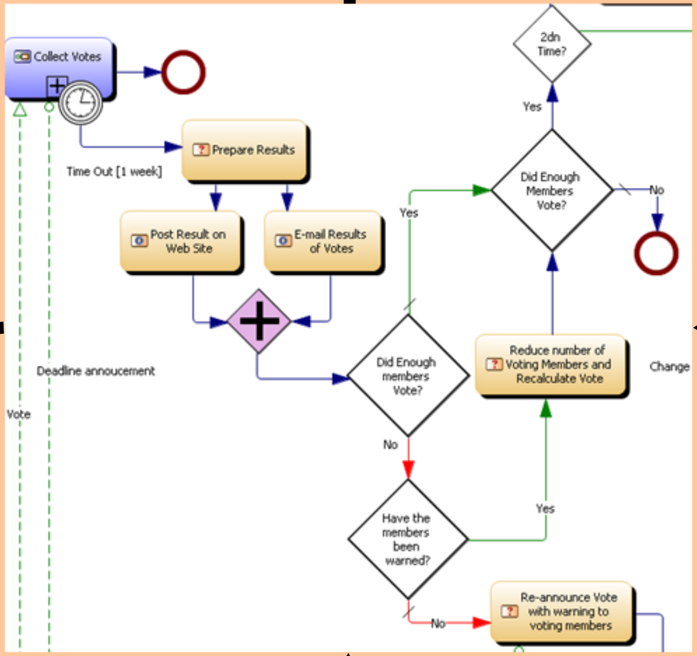


Some Well-known MDSE Concepts

Code generation Generative programming

Model

Code



```
import com.lauchenauer.istockhelper.
       com.lauchenauer.lib.ui.Vertic
       com.lauchenauer.lib.util.Brow

public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;

  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);

    initUI();
  }

  protected void initUI() {
    setSize(440, 600);

    Container cont = getContent
    JPanel p =
    p.set
```

Model Query

Model Refactoring

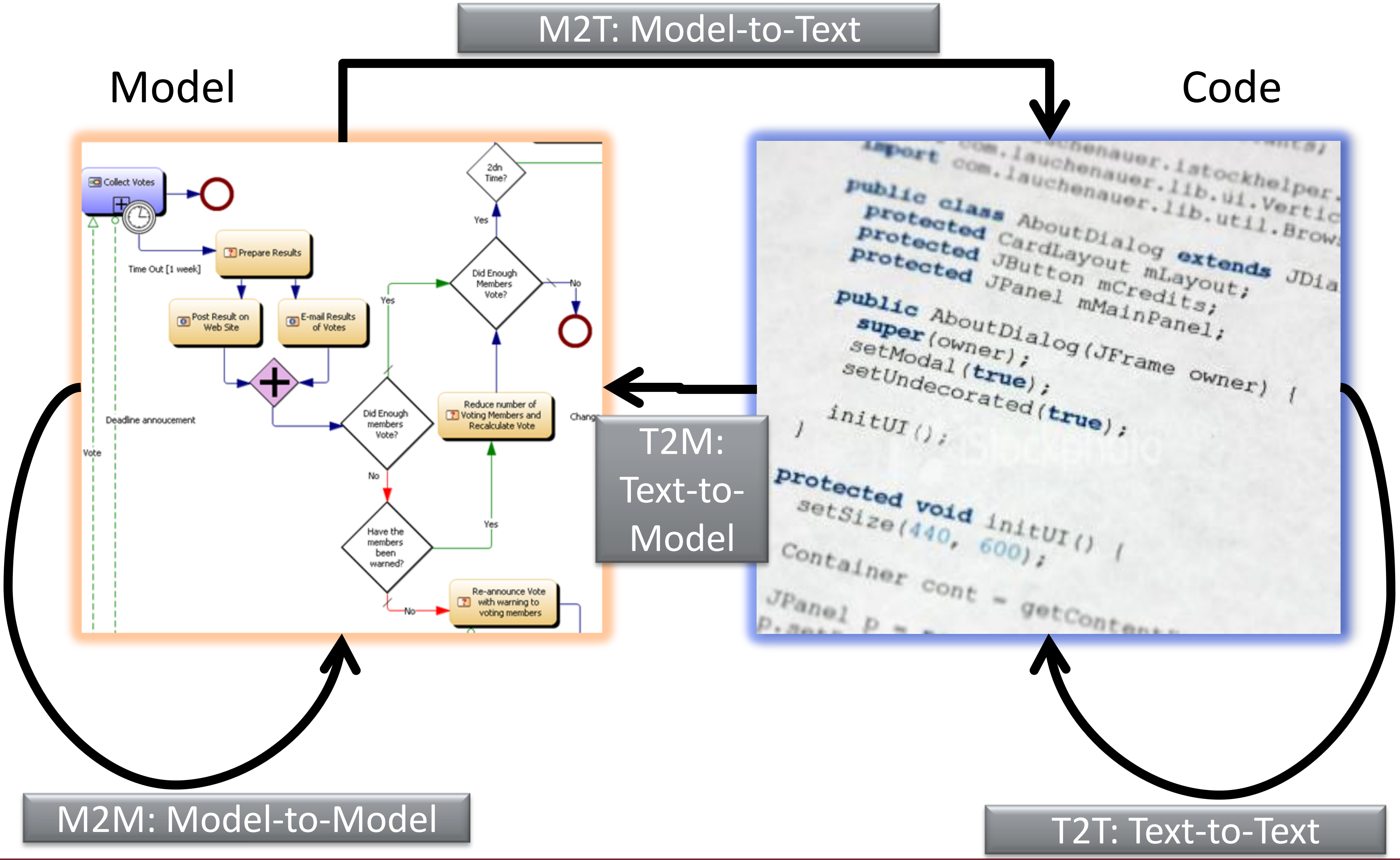
Re-engineering

Program comprehension

Query

Refactoring

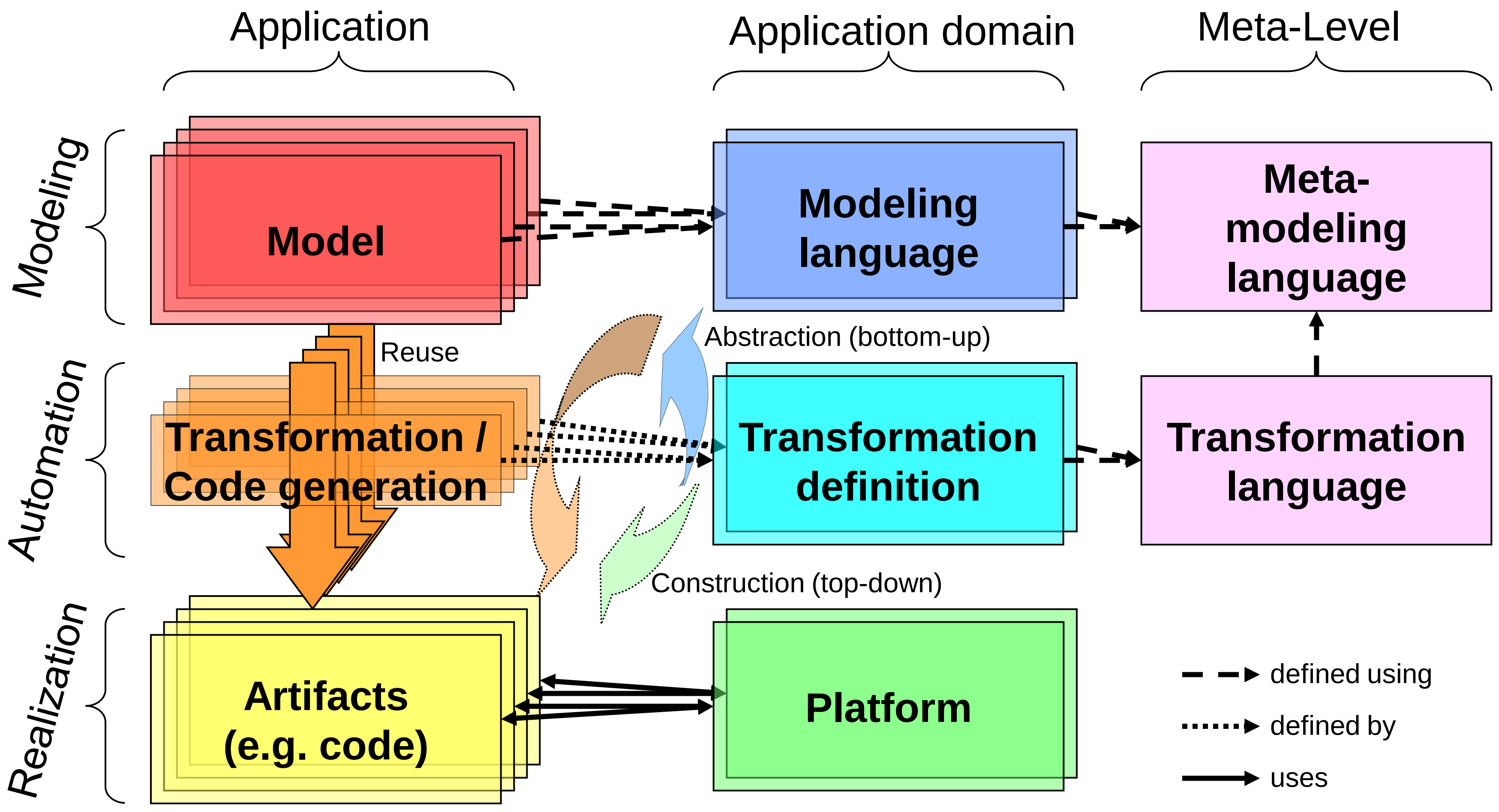
A Classification of Transformations



```
import com.lauchenauer.iatockhelper.
       com.lauchenauer.lib.ui.Vertic
       com.lauchenauer.lib.util.Brow
public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;
 public AboutDialog(JFrame owner) {
   super(owner);
   setModal(true);
   setUndecorated(true);
   initUI();
 }
 protected void initUI() {
   setSize(440, 600);
   Container cont = getContent
   JPanel p =
```

Concepts

Model Engineering basic architecture



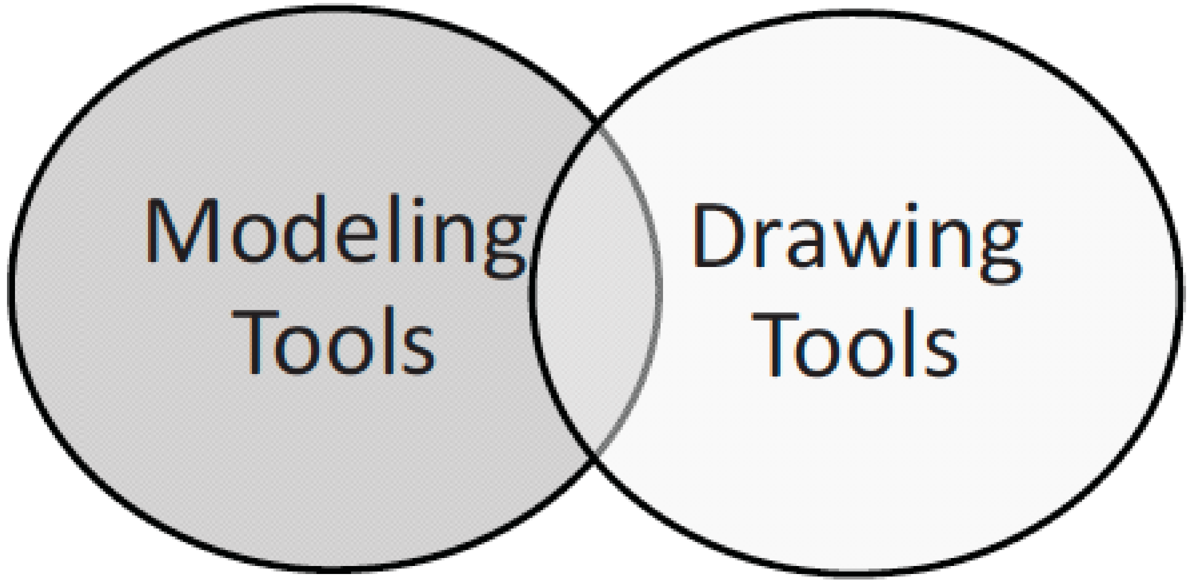
Concepts

Consequences or Preconditions

- Modified **development process**
  - Two levels of development – application and infrastructure
    - Infrastructure development involves modeling language, platform (e.g. framework) and transformation definition
    - Application development only involves modeling – efficient reuse of the infrastructure(s)
  - Strongly simplified application development
    - Automatic code generation replaces programmer
    - Working on the code level (implementation, testing, maintenance) becomes unnecessary
    - *Under which conditions is this realistic … or just futuristic?*
- New **development tools**
  - Tools for language definition, in particular meta modeling
  - Editor and engine for model transformations
  - Customizable tools like model editors, repositories, simulation, verification, and testing tools

Tool support

- Drawing vs. modeling



Motivations for MDSD

Traditional motivations for MDSE

Principles and objectives

- **Abstraction** from specific realization technologies
  - Requires modeling languages, which do not hold specific concepts of realization technologies (e.g., Java EJB)
  - Improved **portability** of software to new/changing technologies – model once, build everywhere
  - **Interoperability** between different technologies can be automated (so called Technology Bridges)

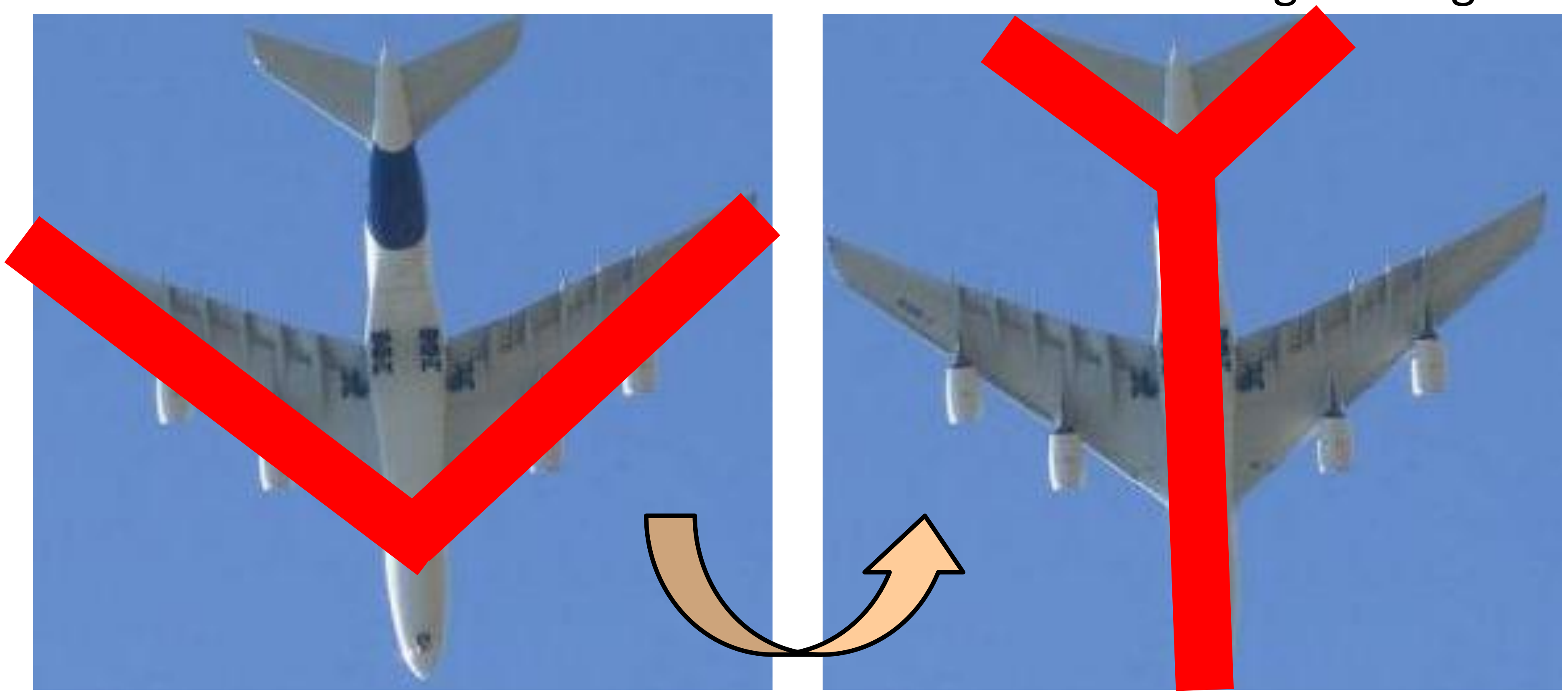
- **Automated code generation** from abstract models
  - e.g., generation of Java-APIs, XML Schemas, etc. from UML
  - Requires expressive und precise models
  - Increased **productivity** and **efficiency** (models stay up-to-date)

- **Separate development** of application and infrastructure
  - Separation of application-code and infrastructure-code (e.g. Application Framework) increases **reusability**
  - **Flexible** development cycles as well as **different development roles possible**

Model-Driven Engineering of Critical Systems

Traditional V-Model

Model-Driven Engineering



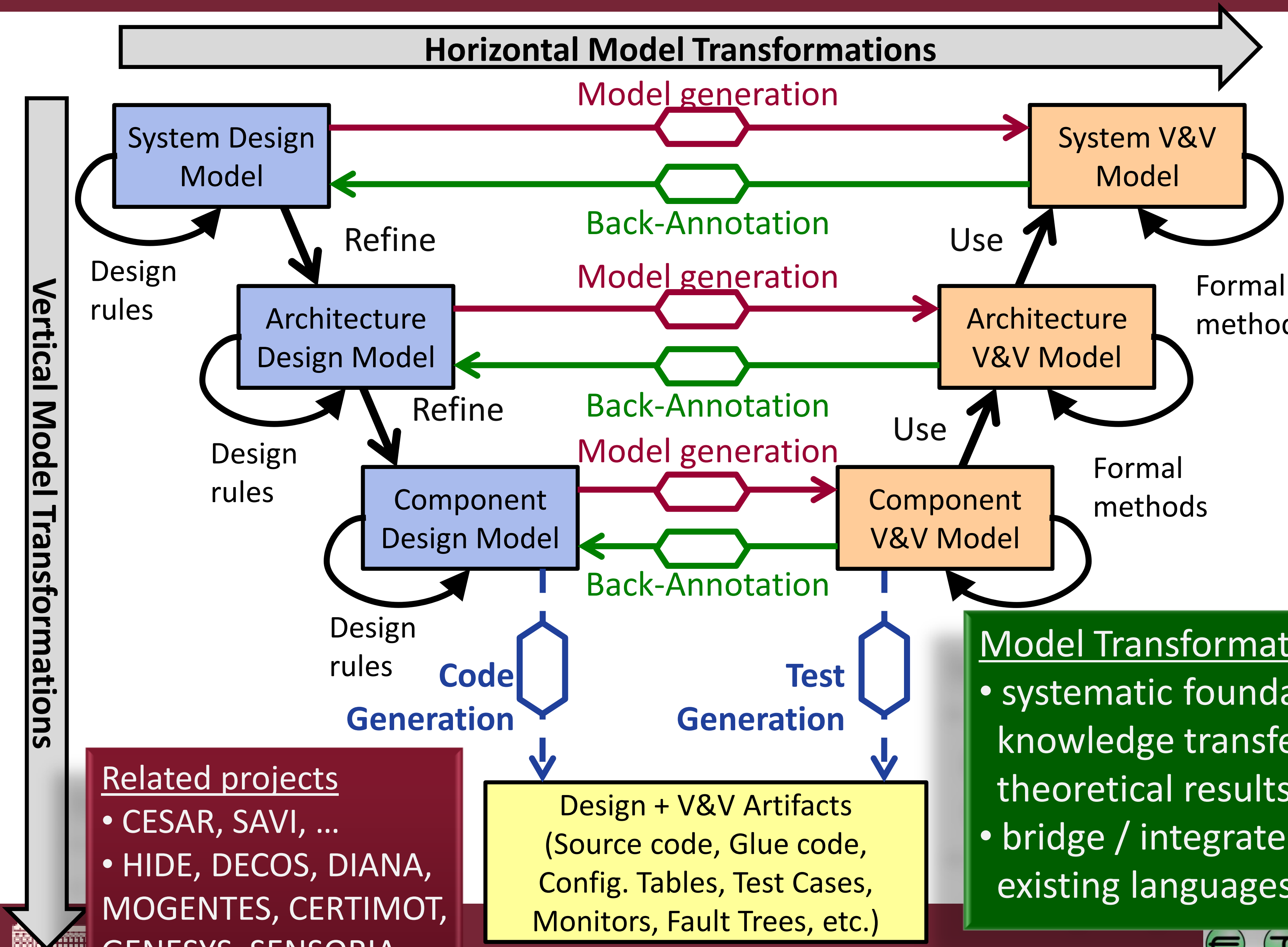
- DO-178B/C: Software Considerations in Airborne Systems and Equipment Certification (RTCA, EUROCAE)
- Steven P. Miller: Certification Issues in Model Based Development (Rockwell Collins)

<u>Main ideas of MDE</u>

- early validation of system models
- automatic source code generation

➔ quality++ tools ++ development cost--

Models and Transformations in Critical Systems

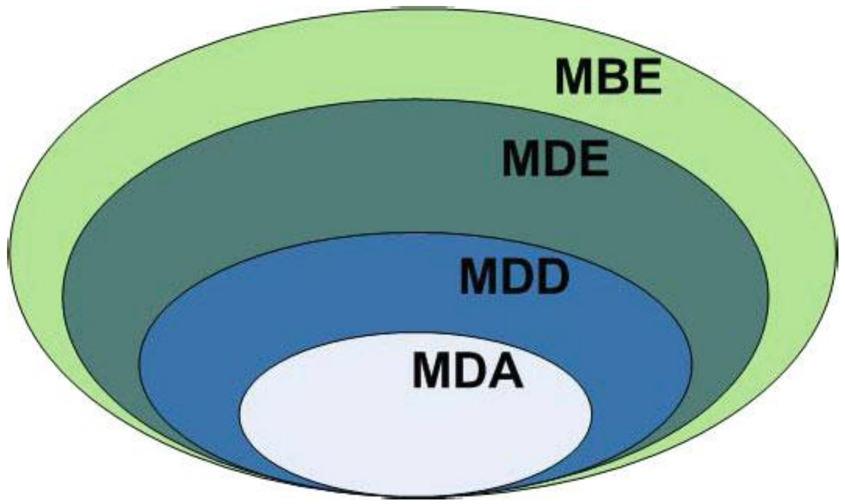


Related projects
- CESAR, SAVI, …
- HIDE, DECOS, DIANA, MOGENTES, CERTIMOT, GENESYS, SENSORIA

Model Transformations
- systematic foundation of knowledge transfer: theoretical results→tools
- bridge / integrate existing languages&tools

MDA = Model-Driven Architecture

The MD* Jungle of Acronyms



- **Model-Driven Development (MDD)** is a development paradigm that uses models as the primary artifact of the development process.
- **Model-Driven Architecture (MDA)** is the particular vision of MDD proposed by the Object Management Group (OMG)
- **Model-Driven Engineering (MDE)** is a superset of MDD because it goes beyond of the pure development
- **Model-Based Engineering** (or "model-based development") (**MBE**) is a softer version of ME, where models do not "drive" the process.

The MDA Approach

Goals

- **Interoperability** through Platform Independent Models
  - Standardization initiative of the Object Management Group (**OMG**), based on OMG Standards, particularly **UML**
  - Counterpart to CORBA on the modeling level: interoperability between different platforms
  - Applications which can be installed on different platforms → portability, no problems with changing technologies, integration of different platforms, etc.
- **Modifications to the basic architecture**
  - Segmentation of the model level
    - **Platform Independent** Models (PIM): valid for a set of (similar) platforms
    - **Platform Specific** Models (PSM): special adjustments for one specific platform
  - Requires model-to-model transformation (PIM-PSM; compare QVT) and model-to-code transformation (PSM-Code)
  - Platform development is not taken into consideration – in general industry standards like J2EE, .NET, CORBA are considered as platforms

[www.omg.org/mda/]

Modeling Levels

CIM, PIM, PSM

- **Computation independent (CIM)**: describe requirements and needs at a very abstract level, without any reference to implementation aspects (e.g., description of user requirements or business objectives);

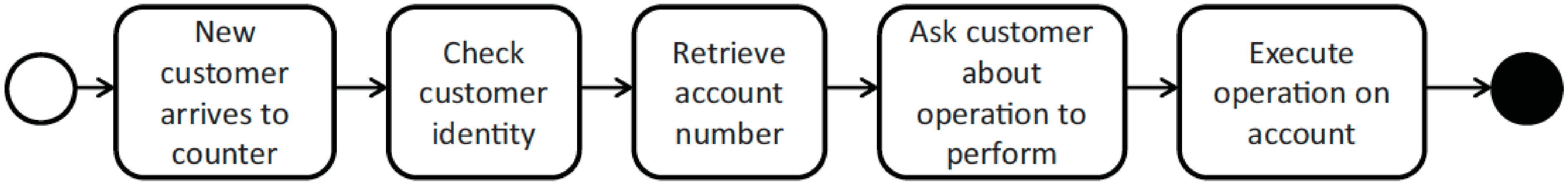
- **Platform independent (PIM)**: define the behavior of the systems in terms of stored data and performed algorithms, without any technical or technological details;

- **Platform-specific (PSM)**: define all the technological aspects in detail.

Modeling levels

MDA Computation Independent Model (CIM)

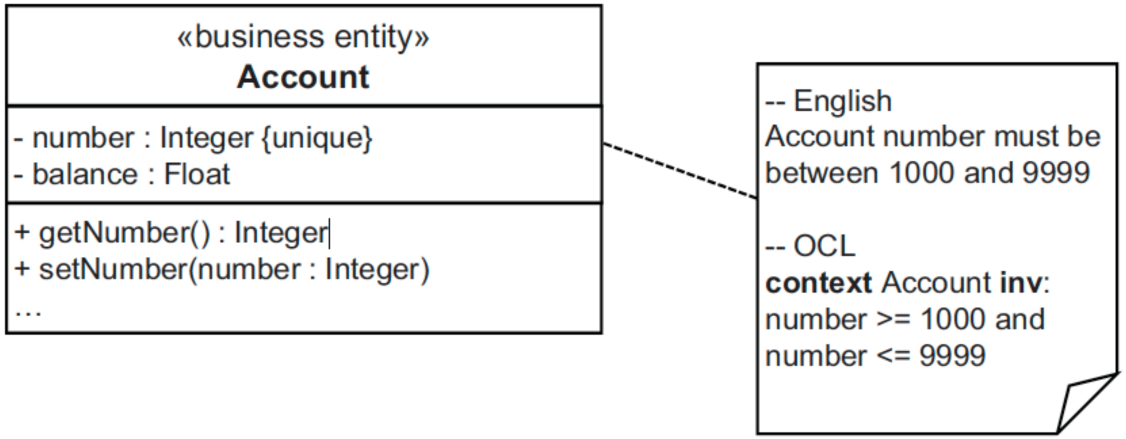
- E.g., business process


New customer arrives to counter
Check customer identity
Retrieve account number
Ask customer about operation to perform
Execute operation on account

Modeling levels

MDA Platform Independent Model (PIM)

- specification of structure and behaviour of a system, abstracted from technologicical details

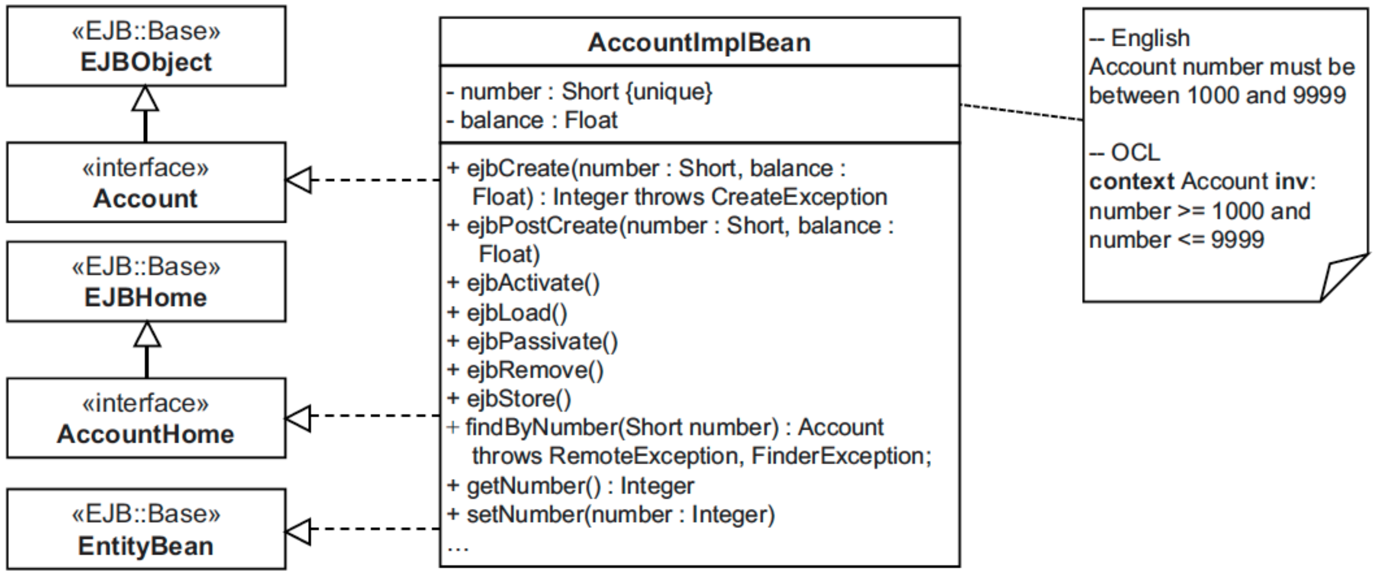


- Using the UML(optional)

- Abstraction of structure and behaviour of a system with the PIM simplifies the following:
  - Validation for correctness of the model
  - Create implementations on different platforms
  - Tool support during implementation

Modeling levels

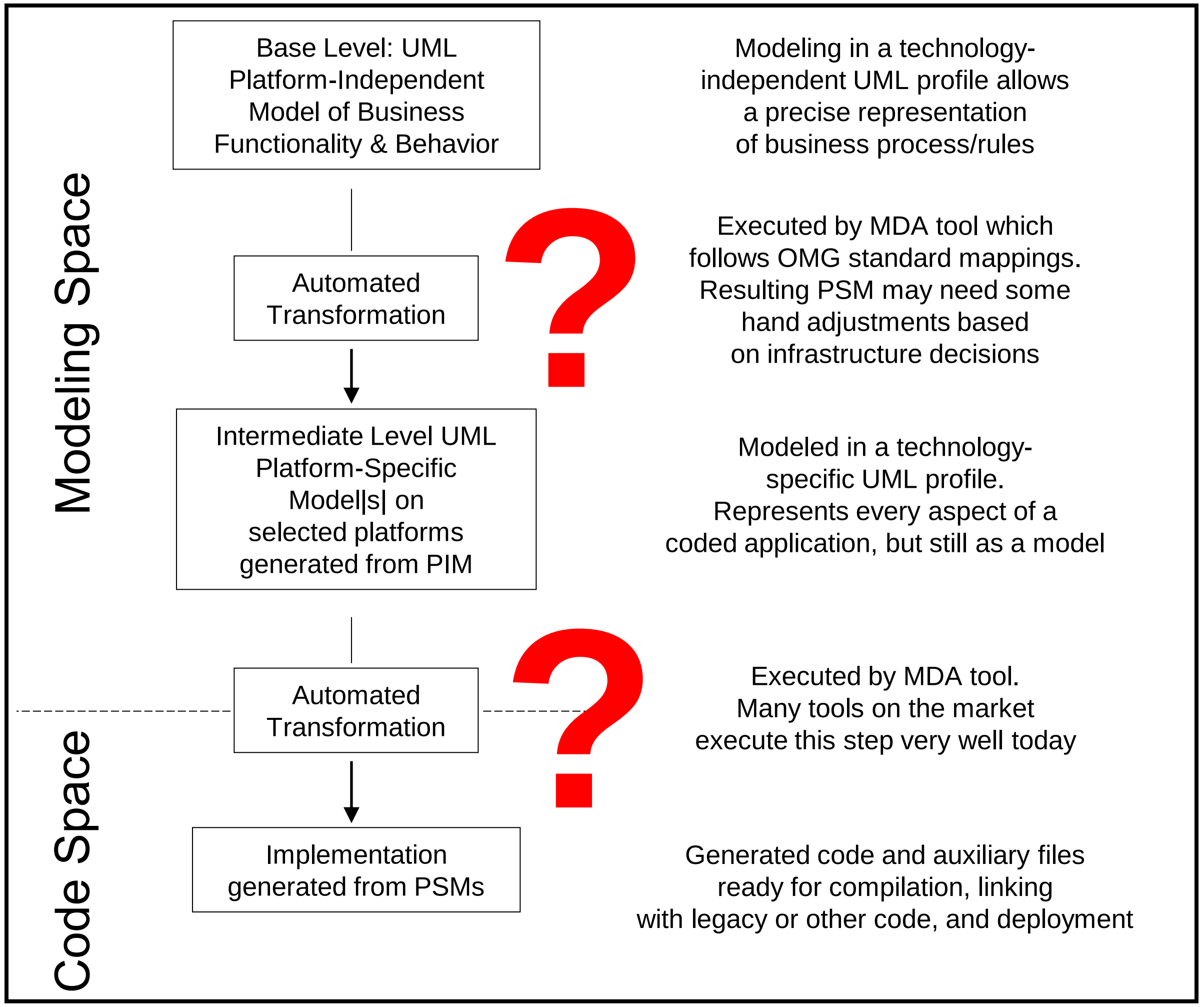
MDA Platform Specific Model (PSM)


«EJB::Base» EJBObject
«interface» Account
«EJB::Base» EJBHome
«interface» AccountHome
«EJB::Base» EntityBean
AccountImplBean
- number : Short {unique}
- balance : Float
+ ejbCreate(number : Short, balance : Float) : Integer throws CreateException
+ ejbPostCreate(number : Short, balance : Float)
+ ejbActivate()
+ ejbLoad()
+ ejbPassivate()
+ ejbRemove()
+ ejbStore()
+ findByNumber(Short number) : Account throws RemoteException, FinderException;
+ getNumber() : Integer
+ setNumber(number : Integer)
...
-- English
Account number must be between 1000 and 9999
-- OCL
context Account inv:
number >= 1000 and number <= 9999

- Specifies how the functionality described in the PIM is realized on a certain platform
- Using a UML-Profile for the selected platform, e.g., EJB

The MDA Approach

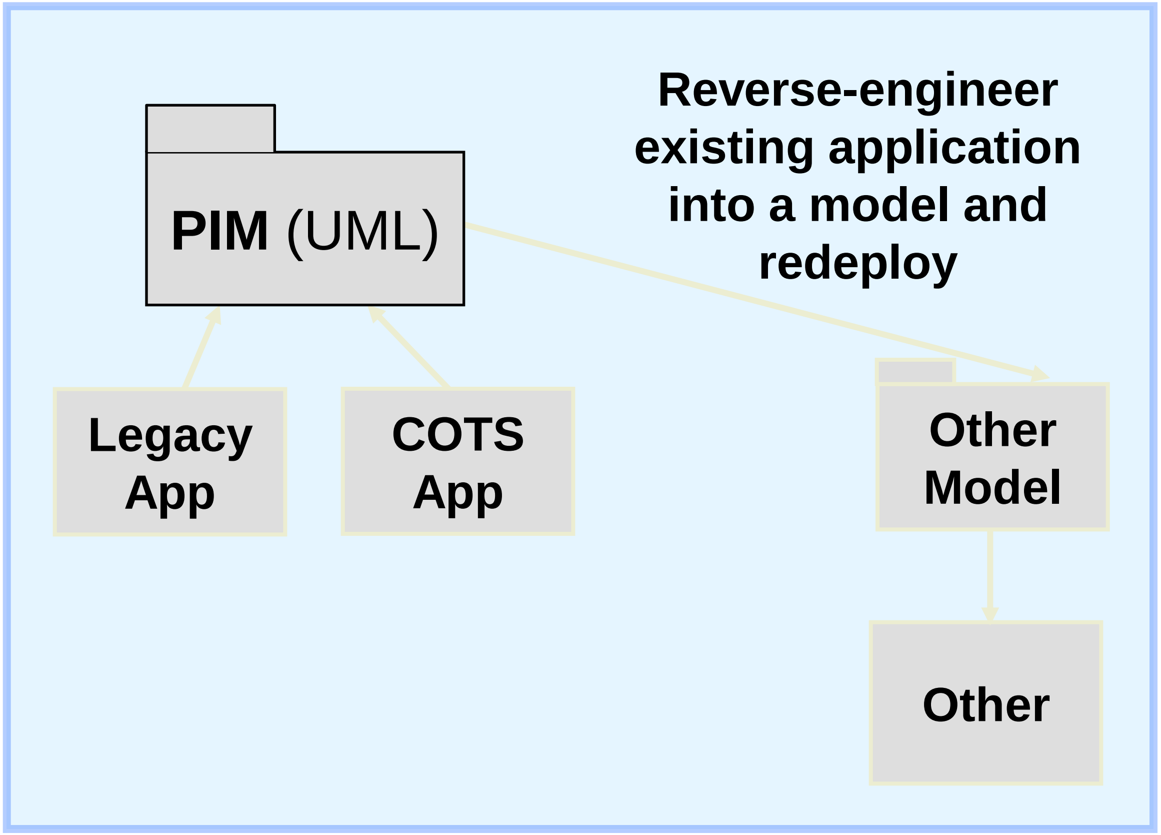
MDA development cycle


Modeling Space
Code Space
Base Level: UML Platform-Independent Model of Business Functionality & Behavior
Automated Transformation
Intermediate Level UML Platform-Specific Model|s| on selected platforms generated from PIM
Automated Transformation
Implementation generated from PSMs
?
?
Modeling in a technology-independent UML profile allows a precise representation of business process/rules
Executed by MDA tool which follows OMG standard mappings. Resulting PSM may need some hand adjustments based on infrastructure decisions
Modeled in a technology-specific UML profile. Represents every aspect of a coded application, but still as a model
Executed by MDA tool. Many tools on the market execute this step very well today
Generated code and auxiliary files ready for compilation, linking with legacy or other code, and deployment

Approaches

MDA Reverse Engineering / Roundtrip Engineering

- Re-integration onto new platforms via Reverse Engineering of an existing application into a PIM und subsequent code generation
- MDA tools for Reverse Engineering automate the model construction from existing code



PIM (UML)
Reverse-engineer existing application into a model and redeploy
Legacy App
COTS App
Other Model
Other

Approaches

Excursus: OMG Standards

- CORBA - Common Object Request Broker Architecture
  - Language- and platform-neutral interoperability standard (similar to WSDL, SOAP and UDDI)
- UML - Unified Modeling Language
  - Standardized modeling language, industry standard
- CWM - Common Warehouse Metamodel
  - Integrated modeling language for Data Warehouses
- MOF – Meta Object Facility
  - A standard for metamodels and model repositories
- XMI - XML Metadata Interchange
  - XML-based exchange of models
- QVT – Queries/Views/Transformations
  - Standard language for Model-to-Model transformations

Summary

- MDSE = Models + Languages + Transformations
- Motivation
  - Early validation of design
  - Automated generation of design artifacts
  - + Interoperability, Productivity, Abstraction, Reuse
- MDA = Model Driven Architecture
  - 3 modeling levels: CIM + PIM + PSM
  - Automated transformations: PIM ➔ PSM ➔ Code (?)

History of MD*

Approaches

Executable UML

- "CASE with UML"
  - **UML-Subset**: Class Diagram, State Machine, Package/Component Diagram, as well as
  - UML Action Semantic Language (ASL) as programming language
- **Niche product**
  - Several specialized vendors like Kennedy/Carter
  - Mainly used for the development of Embedded Systems
- One **part of the basic architecture** implemented
  - Modeling language is predetermined (**xUML**)
  - Transformation definitions can be adapted or can be established by the user (via ASL)
- **Advantages** compared to trad. CASE tools
  - Standardized modeling language based on the UML
- **Disadvantages** compared to trad. CASE tools
  - Limited extent of the modeling language

[S.J. Mellor, M.J. Balcer: Executable UML: a foundation for model-driven architecture. Addison-Wesley, 2002]

Approaches

MDA with UML

- Problems when using **UML** as PIM/PSM
  - Method bodies?
  - Incomplete diagrams, e.g. missing attributes
  - Inconsistent diagrams
  - *For the usage of the UML in Model Engineering special guidelines have to be defined and adhered to*
- Different requirements to **code generation**
  - get/set methods
  - Serialization or persistence of an object
  - Security features, e.g. Java Security Policy
  - *Using adaptable code generators or PIM-to-PSM transformations*
- **Expressiveness** of the UML
  - UML is mainly suitable for "generic" software platforms like Java, EJB, .NET
  - Lack of support for user interfaces, code, etc.
  - *MDA tools often use proprietary extensions*

Approaches
MDA

- Many **UML tools** are expanded to MDA tools
  - UML profiles and code generators
  - Stage of development partly still similar to CASE: proprietary UML profiles and transformations, limited adaptability
- **Advantages** of MDA
  - Standardization of the Meta-Level
  - Separation of platform independent and platform specific models (reuse)
- **Disadvantages** of MDA
  - No special support for the development of the execution platform and the modeling language
  - Modeling language practically limited to UML with profiles
  - Therefore limited code generation (typically no method bodies, user interface)

Approaches

AC-MDSD

- Efficient reuse of architectures
  - Special attention to the efficient reuse of infrastructures/frameworks (= architectures) for a series of applications
  - Specific procedure model
    - Development of a reference application
    - Analysis in individual code, schematically recurring code and generic code (equal for all applications)
    - Extraction of the required modeling concepts and definition of the modeling language, transformations and platform
  - Software support (www.openarchitectureware.org)
- Basic architecture almost completely covered
  - When using UML profiles there is the problem of the method bodies
  - The recommended procedure is to rework these method bodies not in the model but in the generated code
- Advantages compared to MDA
  - Support for platform- and modeling language development
- Disadvantages compared to MDA
  - Platform independence and/or portability not considered

Approaches

MetaCASE/MetaEdit+

- Free configurable CASE
  - Meta modeling for the development of domain-specific modeling languages (**DSL**s)
  - **The focus** is on the ideal support of the **application area**, e.g. mobile-phone application, traffic light pre-emption, digital clock – Intentional Programming
  - Procedural method driven by the DSL development
- Support in particular for the **modeling level**
  - Strong Support for meta modeling, e.g. graphical editors
  - Platform development not assisted specifically, the usage of components and frameworks is recommended
- **Advantages**
  - Domain-specific languages
- **Disadvantages**
  - Tool support only focuses on graphical modeling

[www.metacase.com]

Approaches

Software Factories

- **Series production** of software products
  - Combines the ideas of different approaches (MDA, AC-MDSD, MetaCASE/DSLs) as well as popular SWD-technologies (patterns, components, frameworks)
  - Objective is the automatically processed development of software product series, i.e., a series of applications with the same application area and the same infrastructure
  - The SW-Factory as a marketable product
- Support of the **complete basic architecture**
  - Refinements in particular on the realization level, e.g. deployment
- **Advantages**
  - Comprehensive approach
- **Disadvantages**
  - Approach not clearly delimited (similar MDA)
  - Only little tool support

[J. Greenfield, K. Short: Software Factories. Wiley, 2004]

Eclipse and EMF

- Eclipse Modeling Framework
- Full support for metamodeling and language design
- Fully MD (vs. programming-based tools)
- Used in this course!

Conclusion

Modeling in the last century

- Critical Statements of Software Developers
- »When it comes down to it, the real point of software development is cutting code«
- »Diagrams are, after all, just pretty pictures«
- »No user is going to thank you for pretty pictures; what a user wants is software that executes«

M. Fowler, ”UML Distilled”, 1st edition, Addison Wesley, 1997

Conclusion

Modeling in the new millennium – Much has changed!

- »When it comes down to it, the real point of software development is cutting code«
  - To model or to program, that is not the question!
  - Instead: Talk about the right abstraction level

- »Diagrams are, after all, just pretty pictures«
  - Models are not just notation!
  - Instead: Models have a well-defined syntax in terms of metamodels

- »No user is going to thank you for pretty pictures; what a user wants is software that executes«
  - Models and code are not competitors!
  - Instead: Bridge the gap between design and implementation by model transformations
  - What about the managers?

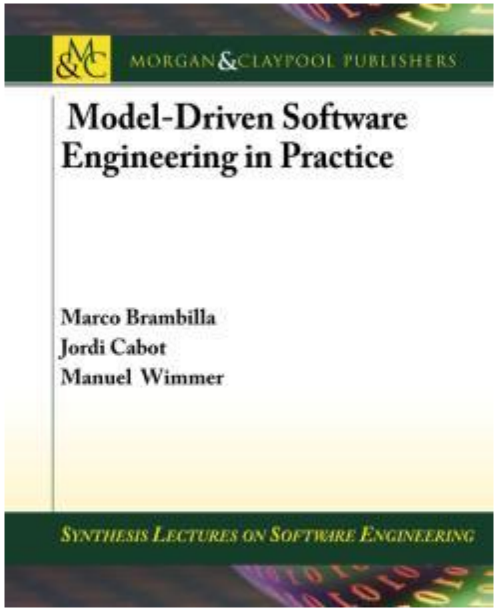
M. Fowler, "UML Distilled", 1st edition, Addison Wesley, 1997
(revisited in 2009)

MORGAN & CLAYPOOL PUBLISHERS

Chapter #2

MDSE PRINCIPLES

Teaching material for the book
Model-Driven Software Engineering in Practice
by Marco Brambilla, Jordi Cabot, Manuel Wimmer.
Morgan & Claypool, USA, 2012.


MORGAN & CLAYPOOL PUBLISHERS

Model-Driven Software Engineering in Practice

Marco Brambilla
Jordi Cabot
Manuel Wimmer

SYNTHESIS LECTURES ON SOFTWARE ENGINEERING