Use of MDE Techniques in ARTIST

9th Symposium and Summer School
On Service-Oriented Computing
Heraklion, Crete, Greece
01-07-15

Outline

Model-Driven Engineering

Models, Metamodels

Model Transformations

MDE in ARTIST

Abstraction and human mind

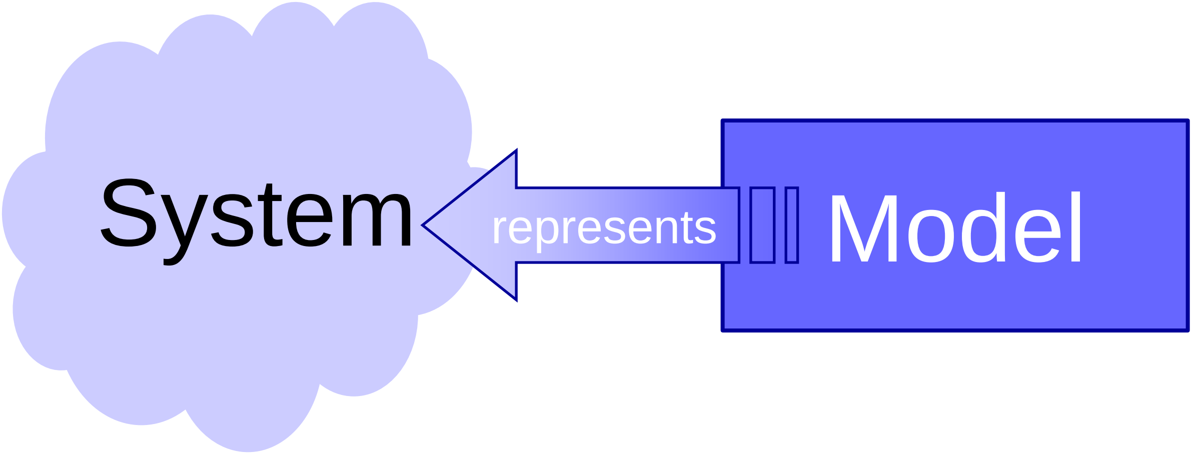
- The human mind continuously re-works reality by applying cognitive processes

- **Abstraction:** capability of finding the commonality in many different observations:
  - generalize specific features of real objects (generalization)
  - classify the objects into coherent clusters (classification)
  - aggregate objects into more complex ones (aggregation)

- **Model:** a simplified or partial representation of reality, defined in order to accomplish a task or to reach an agreement

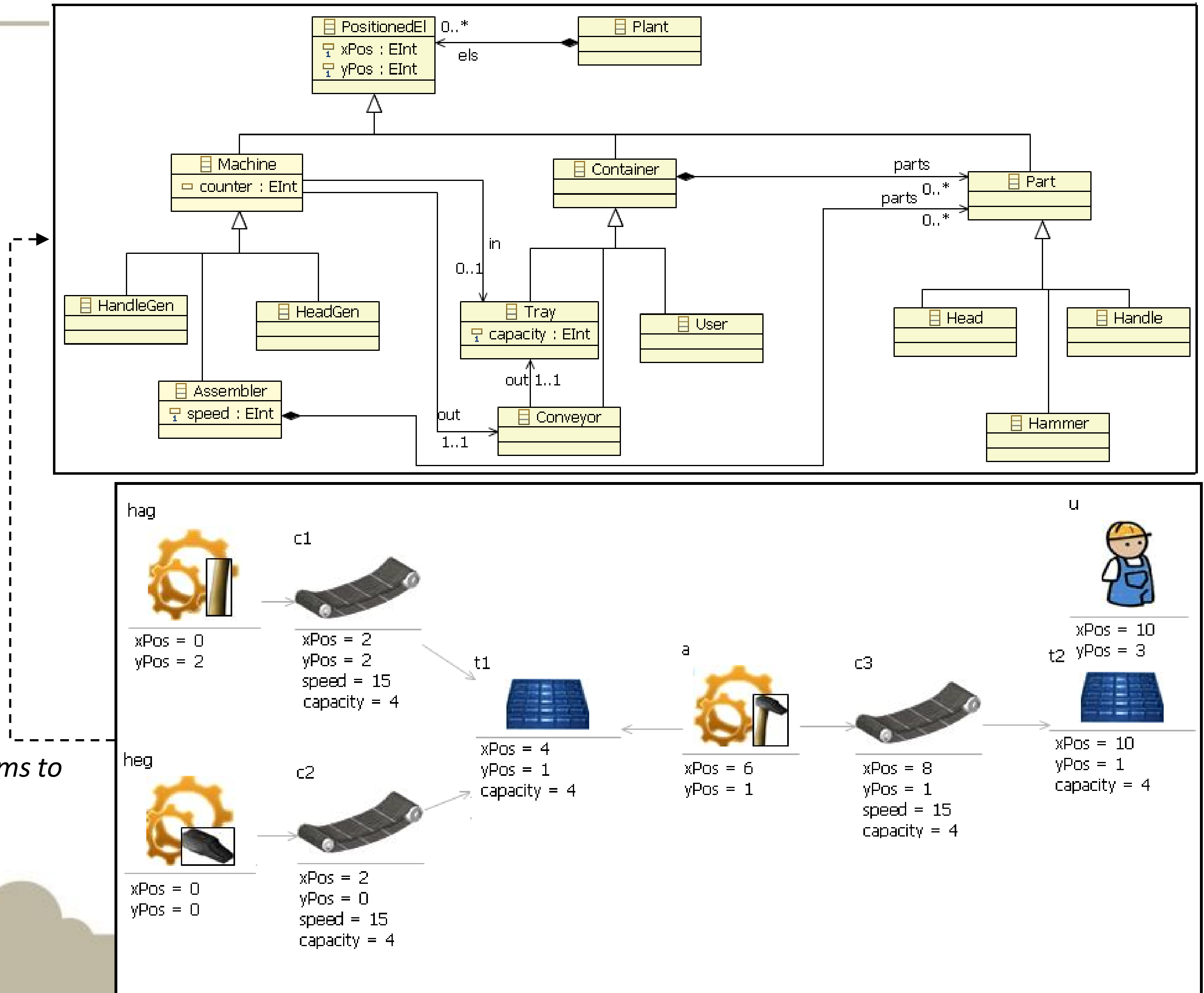
Models

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

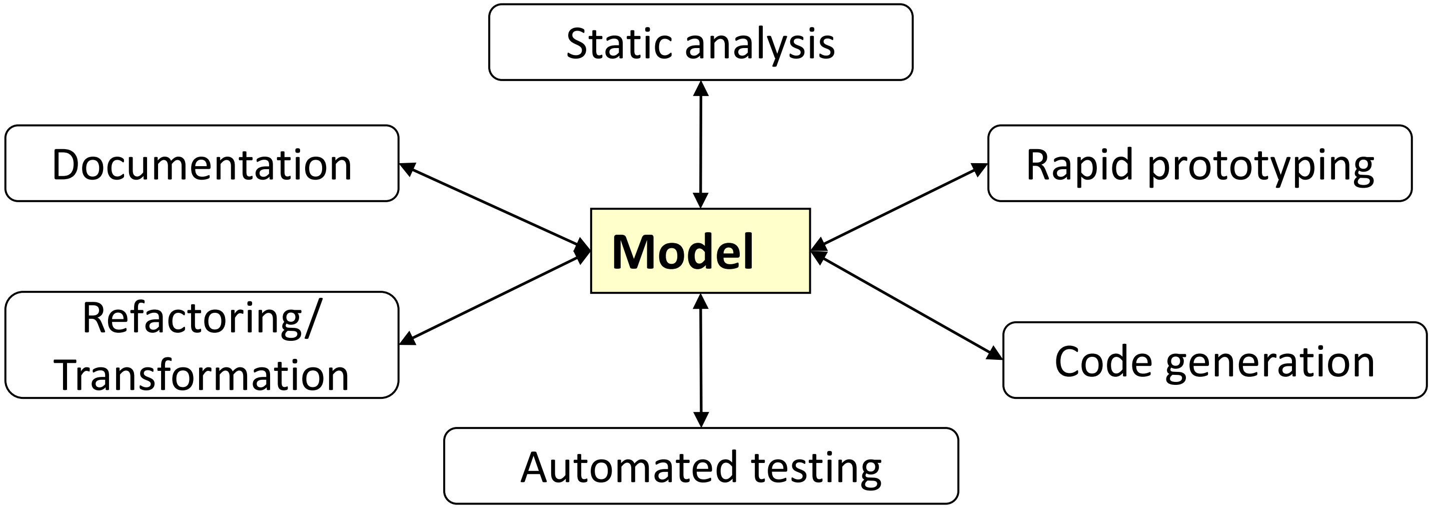
Models and Metamodels



conforms to

Model Engineering

- Model as the **central artifact** of software development



- Related terms
  - Model Driven Engineering (MDE),
  - Model Driven [Software] Development (MDD/MDSD),
  - Model Driven Architecture (MDA)
  - Model Integrated Computing (MIC)

[Illustration by Bernhard Rumpe]

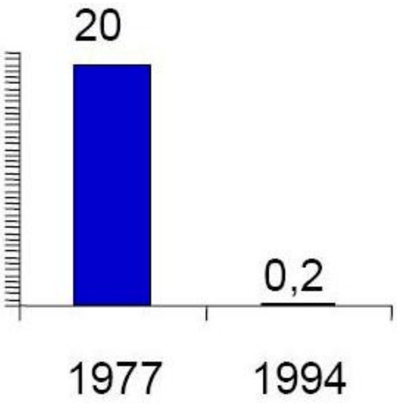
Model Engineering

- Increasing **complexity** of software
  - Increasing basic requirements, e.g., adaptable GUIs, security, network capabilities, …
  - Complex infrastructures, e.g., operating system APIs, language libraries, application frameworks
- Software for **specific devices**
  - Web browser, mobile phone, navigation system, video player, etc.
- **Technological progress …**
  - Integration of different technologies and legacy systems, migration to new technologies
- … leads to **problems** with software development
  - Software finished too late
  - Wrong functionality realized
  - Software is poorly documented/commented
  - and can not be further developed, e.g., when the technical environment changes, business model/ requirements change, etc.

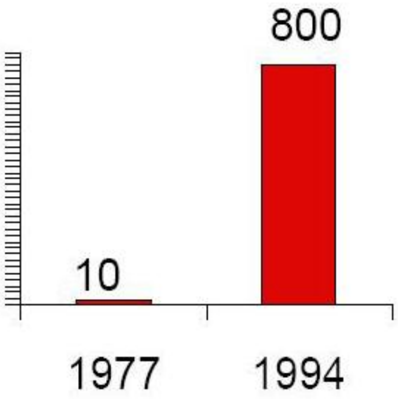
Model Engineering

[Balzert, H.: Lehrbuch der Softwaretechnik: Software-Entwicklung, Spektrum, Akad. Verlag, 1996]

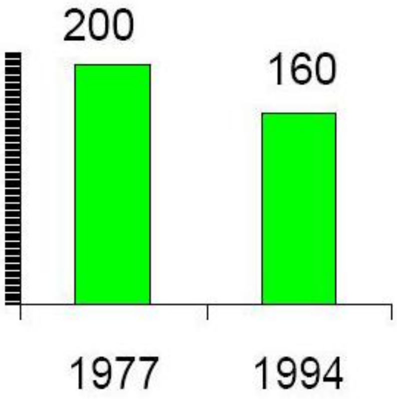
- **Quality problems** in software development



Number of bugs per 1000 LOC



Program size (1000 LOC)



Resulting absolute bug count

Real quality improvements are only possible if the increase in program complexity is **overcompensated** !

(Average values, from Balzert 96)

[Slide by Bernhard Rumpe]

Model Engineering

- **Traditional** usage of models in software development
  - **Communication** with customers and users (requirement specification, prototypes)
  - Support for software design, capturing of the **intention**
  - **Task specification** for programming
  - **Code visualization**, for example in TogetherJ

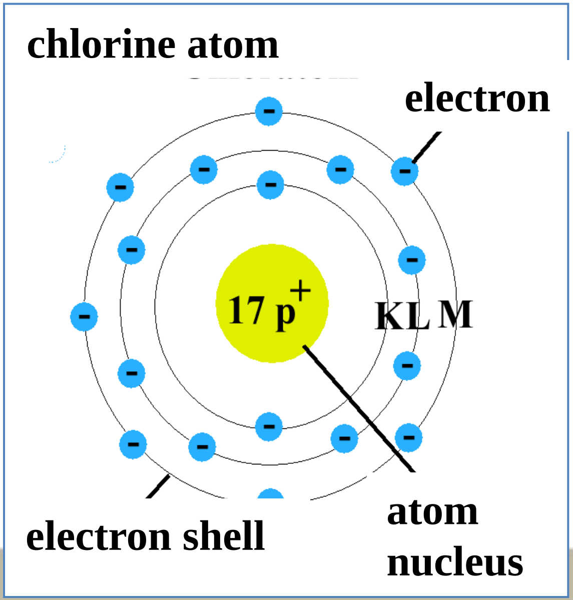
- Clear **difference** to Model Engineering

Models

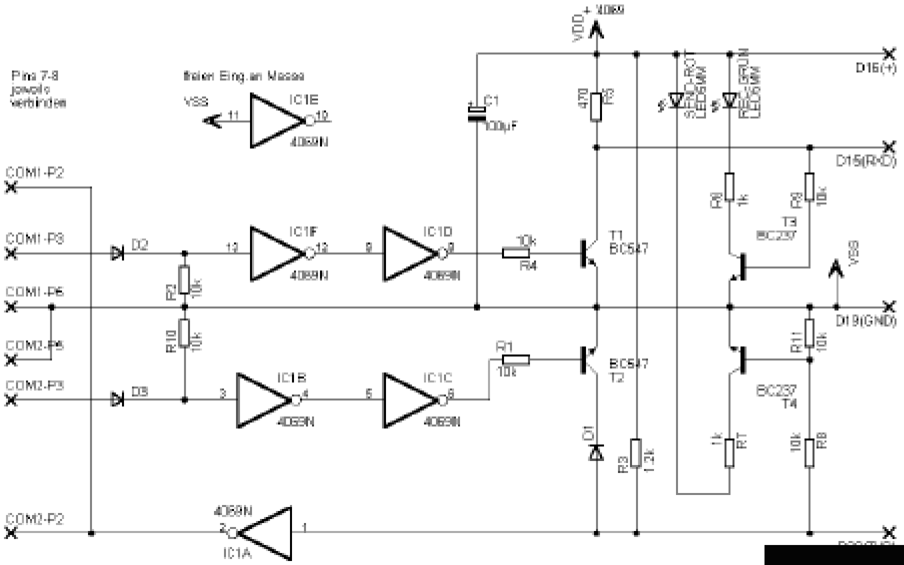
- Do not apply models as long as you have not checked the underlying **simplifications** and evaluated its **practicability**.

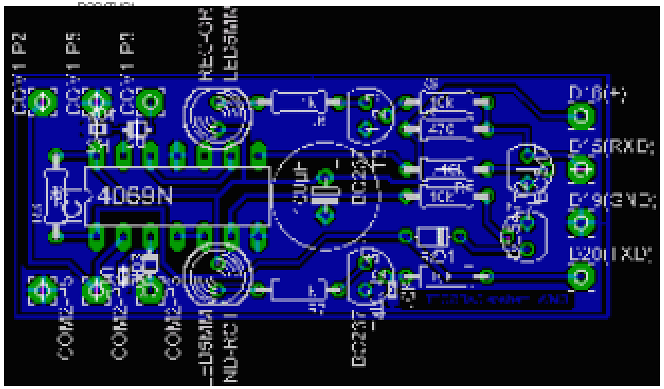
- Never mistake the **model** for the **reality**.
  - Attention: abstraction, abbreviation, approximation, visualization, …

U1 U-Bahn-Linie
S1 S-Bahn-Linie
City Airport Train
Lokalbahn Wien–Baden
Kundenzentrum der Wiener Linien (U3-Station Erdberg)
Informationsstelle der Wiener Linien
Vorverkaufsstelle der Wiener Linien
Flughafen
Park & Ride
U1-Verlängerung (2.9.2006)
U2-Verlängerung (2008)
U2-Verlängerung (2010)



Models

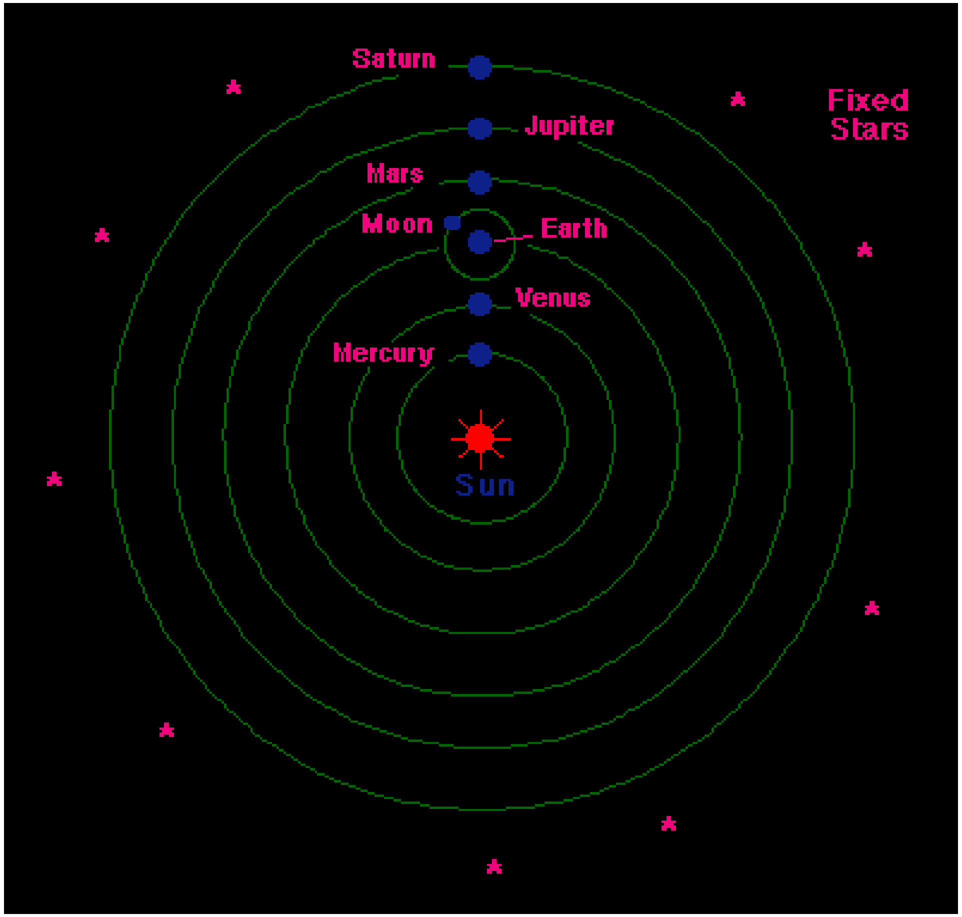




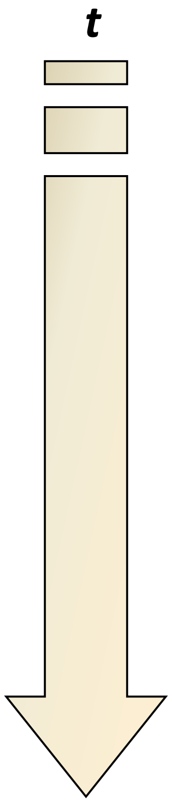
[Slide by Bernhard Rumpe]

Models

- Heliocentric model by Kopernikus



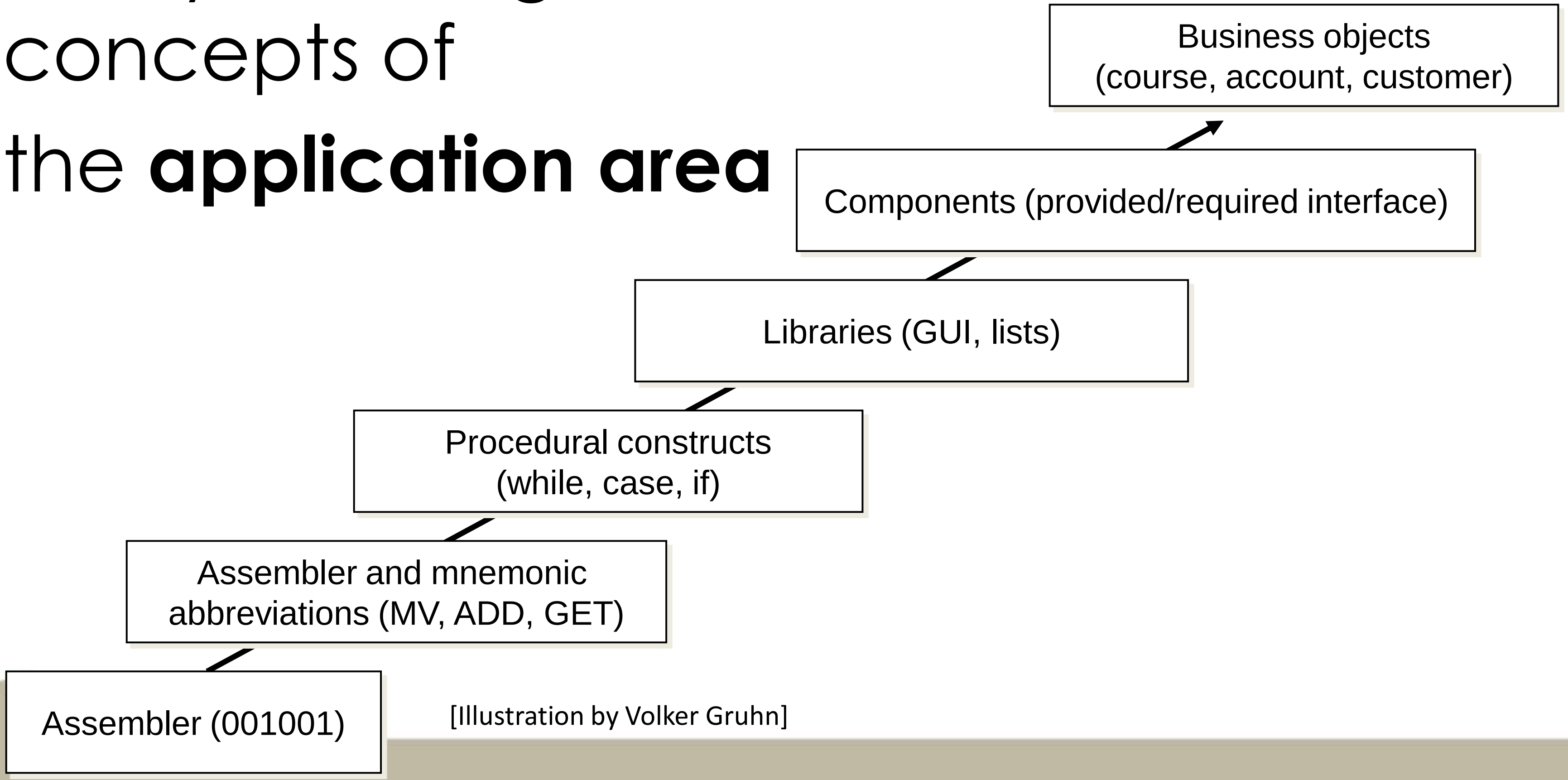
Models



- ***Models as drafts***
  - Communication of ideas and alternatives
  - Objective: modeling per se
- ***Models as guidelines***
  - Design decisions are documented
  - Objective: instructions for implementation
- ***Models as programs***
  - Applications are generated automatically
  - Objective: models are source code and vice versa

Model Engineering

- The **used artifacts of software development** slowly converge to the concepts of the **application area**



Business objects (course, account, customer)

Components (provided/required interface)

Libraries (GUI, lists)

Procedural constructs (while, case, if)

Assembler and mnemonic abbreviations (MV, ADD, GET)

Assembler (001001)

[Illustration by Volker Gruhn]

Modeling Levels

- **Computation independent Model (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 Model (PIM):** define the behavior of the systems in terms of stored data and performed algorithms, without any technical or technological details
- **Platform-specific Model (PSM):** define all the technological aspects in detail.

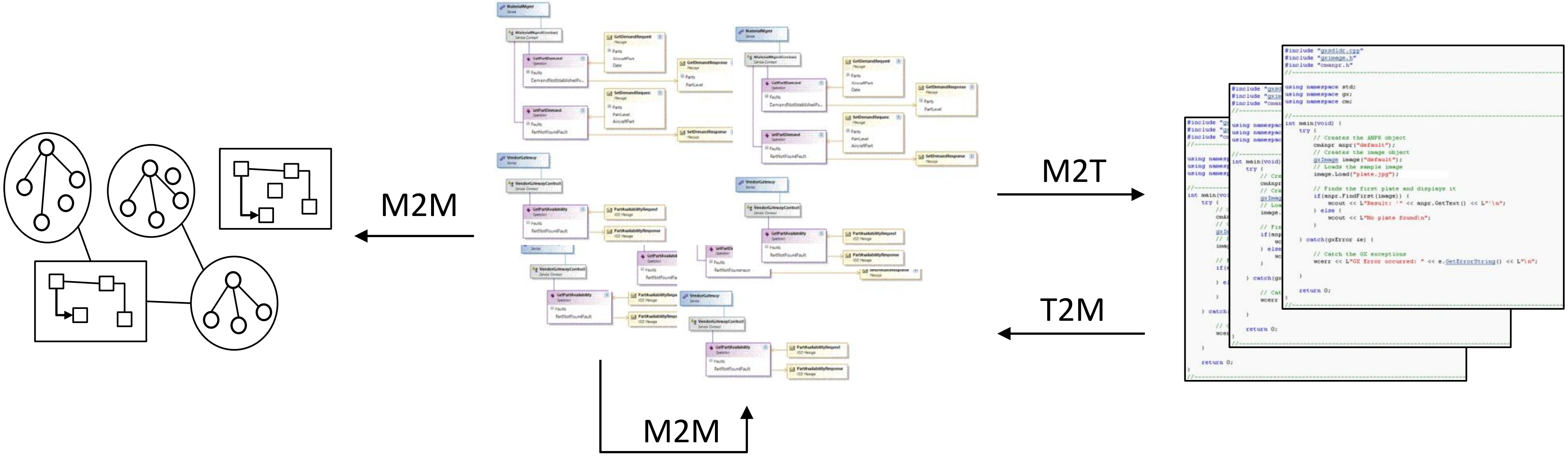
MDE Equation

Models + Transformations = Software

Model Transformations

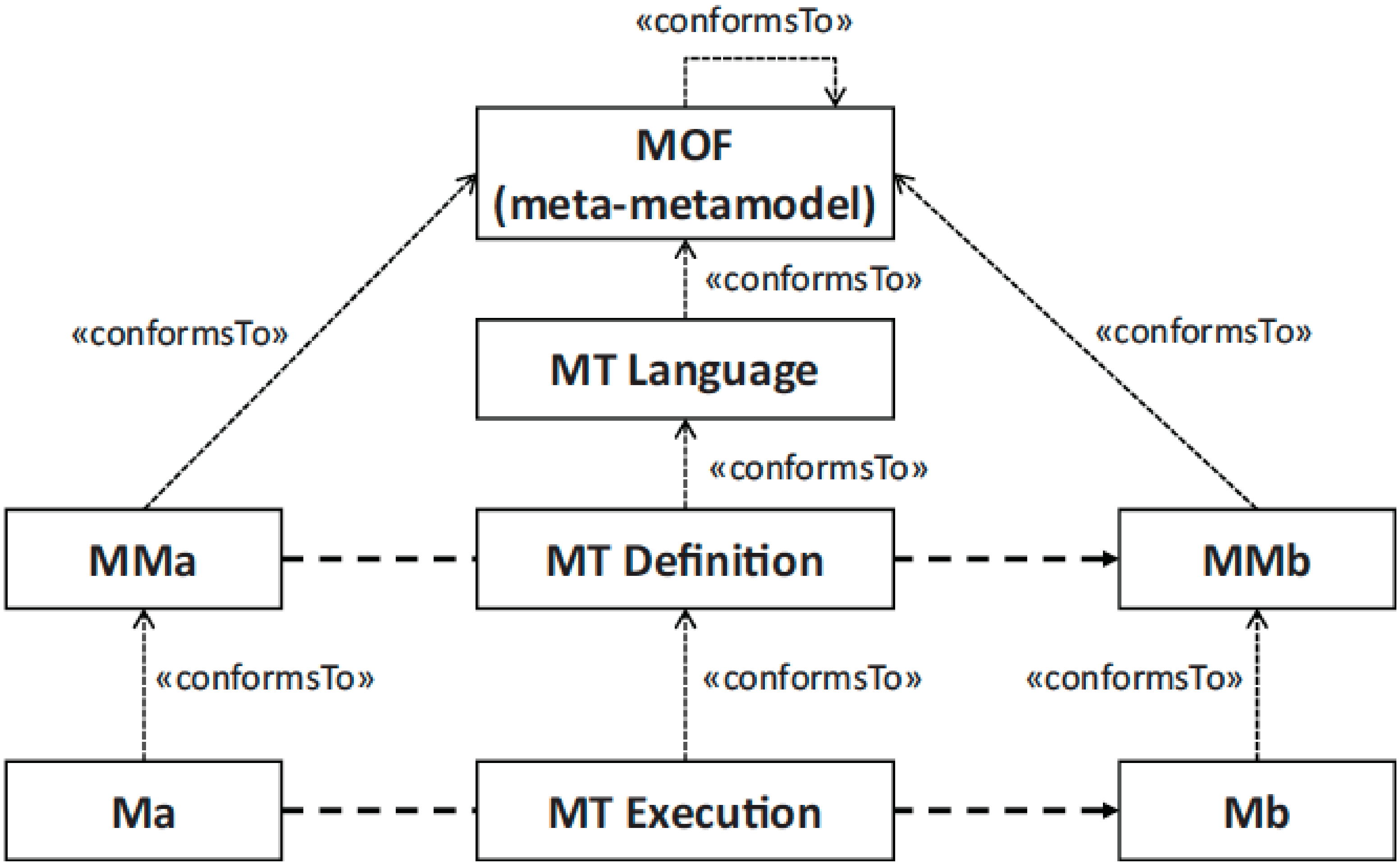
- Transforming items
- MDE provides appropriate languages for defining model transformation rules
- Rules can be written manually from scratch by a developer, or can be defined as a refined specification of an existing one.
- Alternatively, transformations themselves can be produced automatically out of some higher level mapping rules between models
  - defining a mapping between elements of a model to elements of another model (**model mapping or model weaving**)
  - automating the generation of the actual transformation rules through a system that receives as input the two model definitions and the mapping
- Transformations themselves can be seen as models!!

Model Transformations



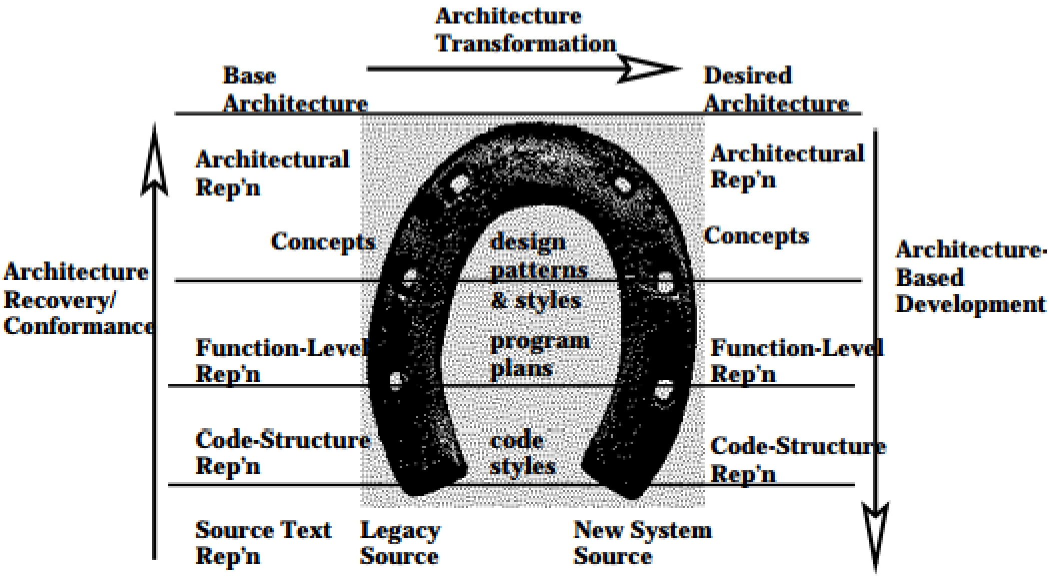
```
#include "gxsdldr.cpp"
#include "gximage.h"
#include "cmanpr.h"
//------------------------------------------------------------------------

using namespace std;
using namespace gx;
using namespace cm;

//------------------------------------------------------------------------
int main(void) {
    try {
        // Creates the ANPR object
        cmAnpr anpr("default");
        // Creates the image object
        gxImage image("default");
        // Loads the sample image
        image.Load("plate.jpg");

        // Finds the first plate and displays it
        if(anpr.FindFirst(image)) {
            wcout << L"Result: '" << anpr.GetText() << L"'\n";
        } else {
            wcout << L"No plate found\n";
        }

    } catch(gxError &e) {

        // Catch the GX exceptions
        wcerr << L"GX Error occurred: " << e.GetErrorString() << L"\n";

    }

    return 0;
}
//------------------------------------------------------------------------
```

Model Transformations



Model-Based Migration Approach in ARTIST

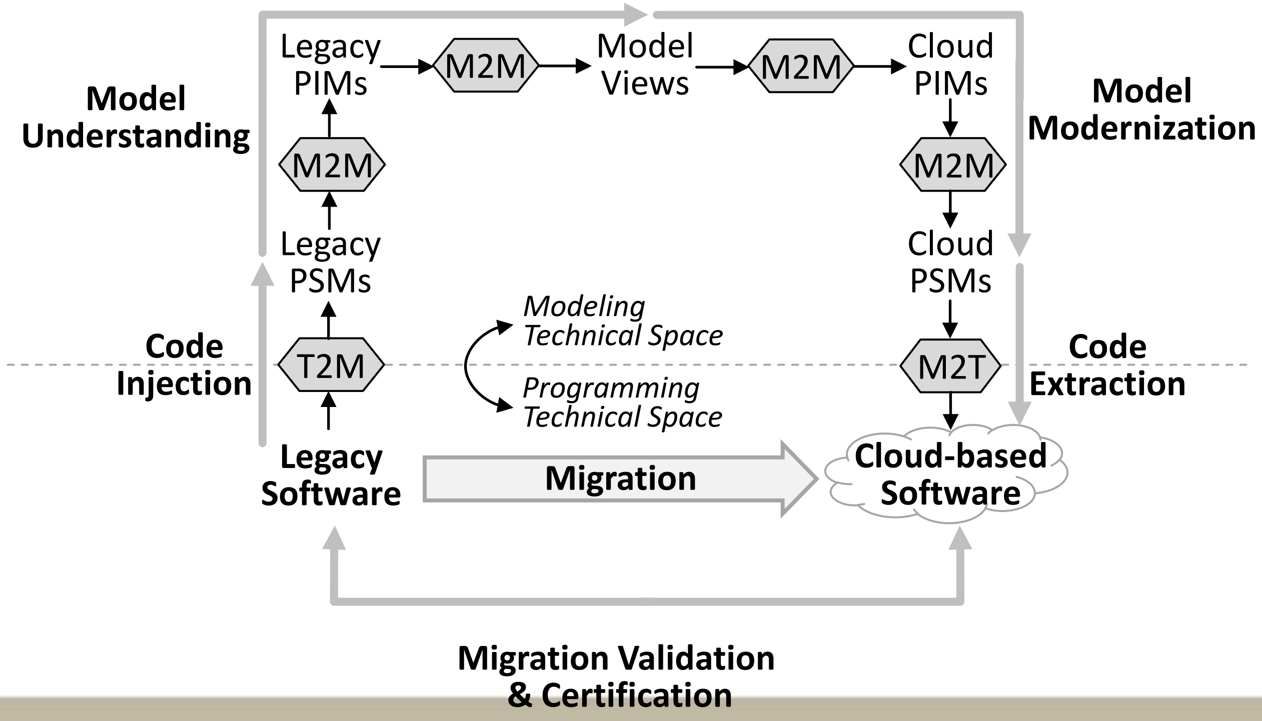
- Inspired from existing reengineering processes, e.g., *Horseshoe Model*
- **Reinterpretation** in the light of advances in Model-based Engineering
  - (Meta-)models and model transformations as a foundation
  - Discover and understand on-premise environments by abstraction(s)
  - Refine abstracted model(s) for selected cloud environments



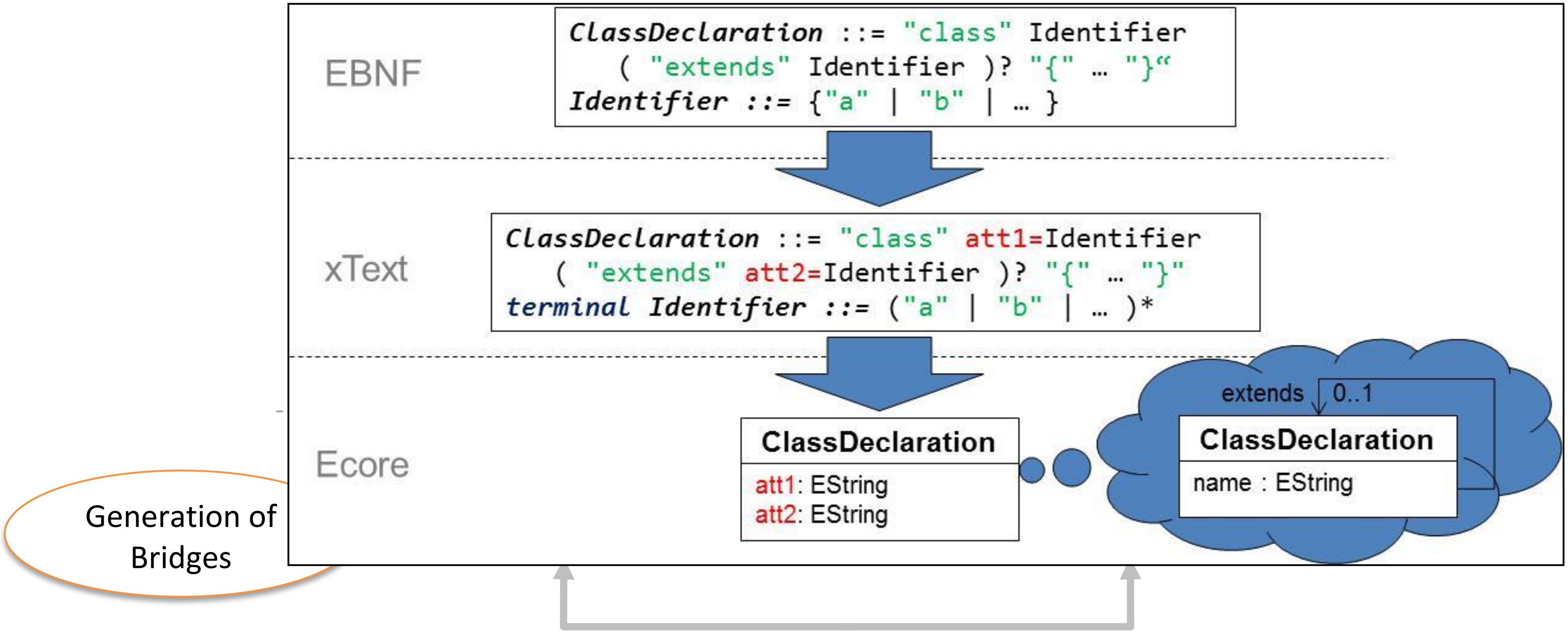
Kazman, R., Woods, S. G., & Carrière, S. J.: Requirements for integrating software architecture and reengineering models: CORUM II. In: Proc. WCRE, 154-163 (1998).
Object Management Group: Architecture Driven Modernization, http://adm.omg.org/
France, R., Rumpe, B.: Modeling for the cloud. SoSyM 9(2), 139–140 (2010)
France, R., Rumpe, B.: The Evolution of Modeling Research Challenges. SoSyM 12(2), 223–225 (2013)

Model-Based Migration Approach in ARTIST

- Inspired from existing reengineering processes, e.g., *Horseshoe Model*
- **Reinterpretation** in the light of advances in Model-based Engineering
  - (Meta-)models and model transformations as a foundation
  - Discover and understand on-premise environments by abstraction(s)
  - Refine abstracted model(s) for selected cloud environments



Model-Based Migration Approach in ARTIST



```
ClassDeclaration ::= "class" Identifier
   ( "extends" Identifier )? "{" … "}“
Identifier ::= {"a" | "b" | … }
```

```
ClassDeclaration ::= "class" att1=Identifier
   ( "extends" att2=Identifier )? "{" … "}"
terminal Identifier ::= ("a" | "b" | … )*
```

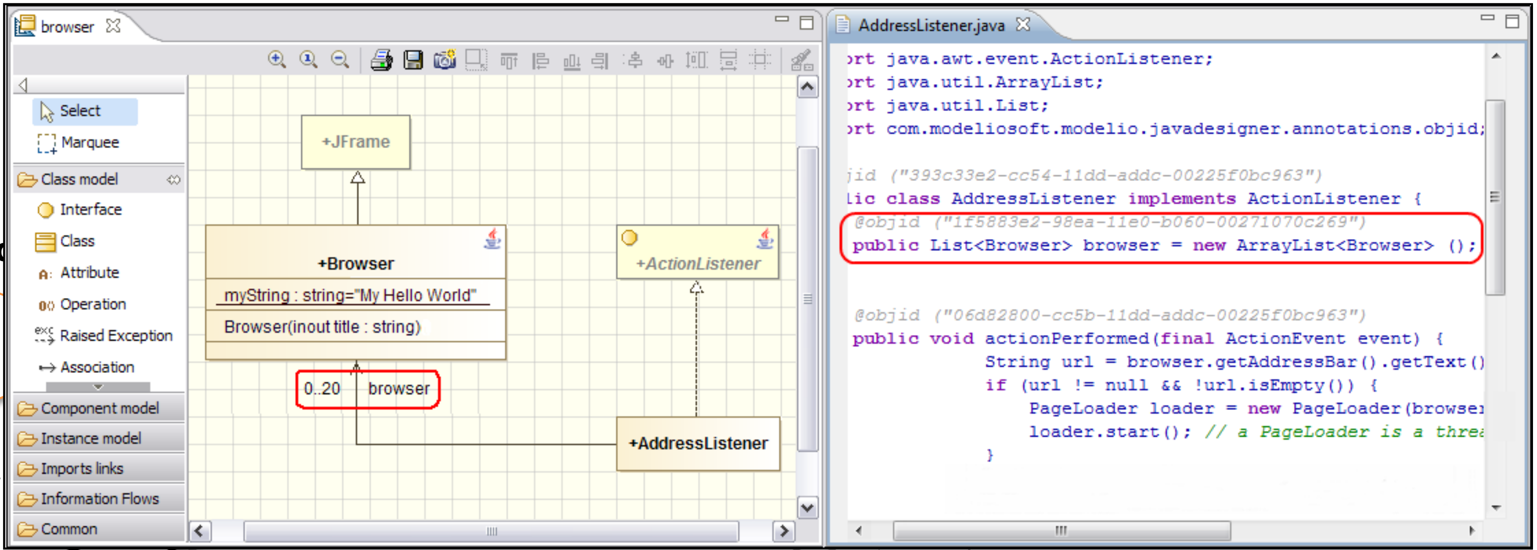
Generation of Bridges

Migration Validation
& Certification

Model-Based Migration Approach in ARTIST

Und

UML Model Extraction



```
ort java.awt.event.ActionListener;
ort java.util.ArrayList;
ort java.util.List;
ort com.modeliosoft.modelio.javadesigner.annotations.objid;

jid ("393c33e2-cc54-11dd-addc-00225f0bc963")
lic class AddressListener implements ActionListener {
 @objid ("1f5883e2-98ea-11e0-b060-00271070c269")
 public List<Browser> browser = new ArrayList<Browser> ();


 @objid ("06d82800-cc5b-11dd-addc-00225f0bc963")
 public void actionPerformed(final ActionEvent event) {
             String url = browser.getAddressBar().getText()
             if (url != null && !url.isEmpty()) {
                 PageLoader loader = new PageLoader(browse
                 loader.start(); // a PageLoader is a threa
             }
```

Generation of Bridges

Legacy Software

Migration

Cloud-based Software

Migration Validation & Certification

Model-Based Migration Approach in ARTIST

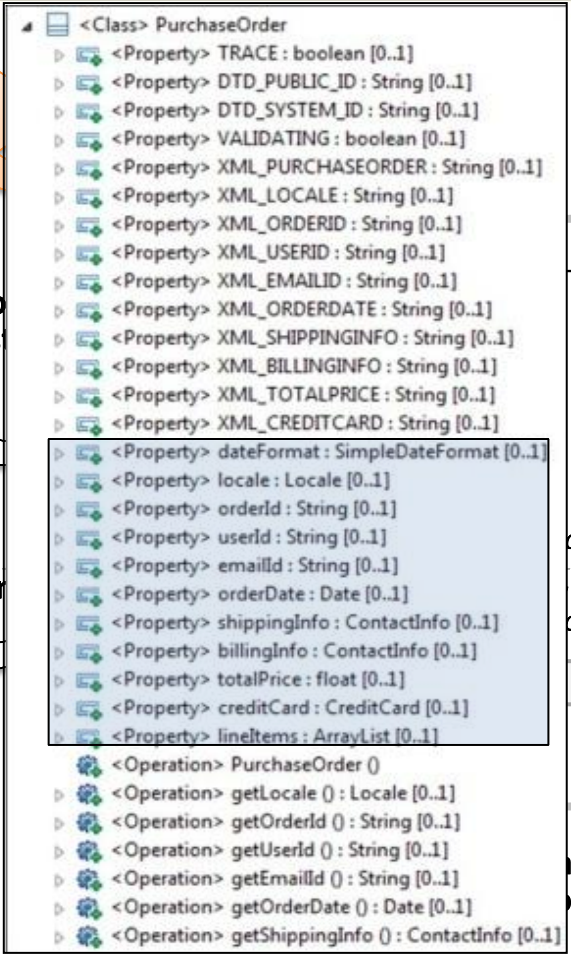
Separation PIM/PDM

Mo
Underst

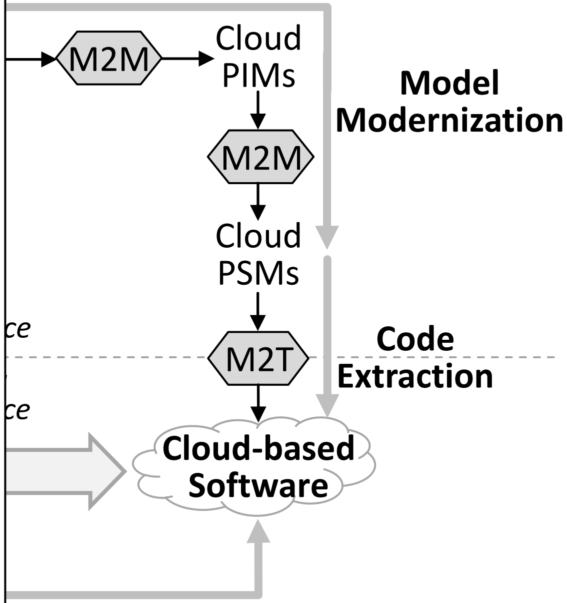
UML Model Extraction

Ir

Generation of Bridges



<Class> PurchaseOrder
<Property> TRACE : boolean [0..1]
<Property> DTD_PUBLIC_ID : String [0..1]
<Property> DTD_SYSTEM_ID : String [0..1]
<Property> VALIDATING : boolean [0..1]
<Property> XML_PURCHASEORDER : String [0..1]
<Property> XML_LOCALE : String [0..1]
<Property> XML_ORDERID : String [0..1]
<Property> XML_USERID : String [0..1]
<Property> XML_EMAILID : String [0..1]
<Property> XML_ORDERDATE : String [0..1]
<Property> XML_SHIPPINGINFO : String [0..1]
<Property> XML_BILLINGINFO : String [0..1]
<Property> XML_TOTALPRICE : String [0..1]
<Property> XML_CREDITCARD : String [0..1]
<Property> dateFormat : SimpleDateFormat [0..1]
<Property> locale : Locale [0..1]
<Property> orderId : String [0..1]
<Property> userId : String [0..1]
<Property> emailId : String [0..1]
<Property> orderDate : Date [0..1]
<Property> shippingInfo : ContactInfo [0..1]
<Property> billingInfo : ContactInfo [0..1]
<Property> totalPrice : float [0..1]
<Property> creditCard : CreditCard [0..1]
<Property> lineItems : ArrayList [0..1]
<Operation> PurchaseOrder ()
<Operation> getLocale () : Locale [0..1]
<Operation> getOrderId () : String [0..1]
<Operation> getUserId () : String [0..1]
<Operation> getEmailId () : String [0..1]
<Operation> getOrderDate () : Date [0..1]
<Operation> getShippingInfo () : ContactInfo [0..1]



Model Modernization

Code Extraction

tion
on

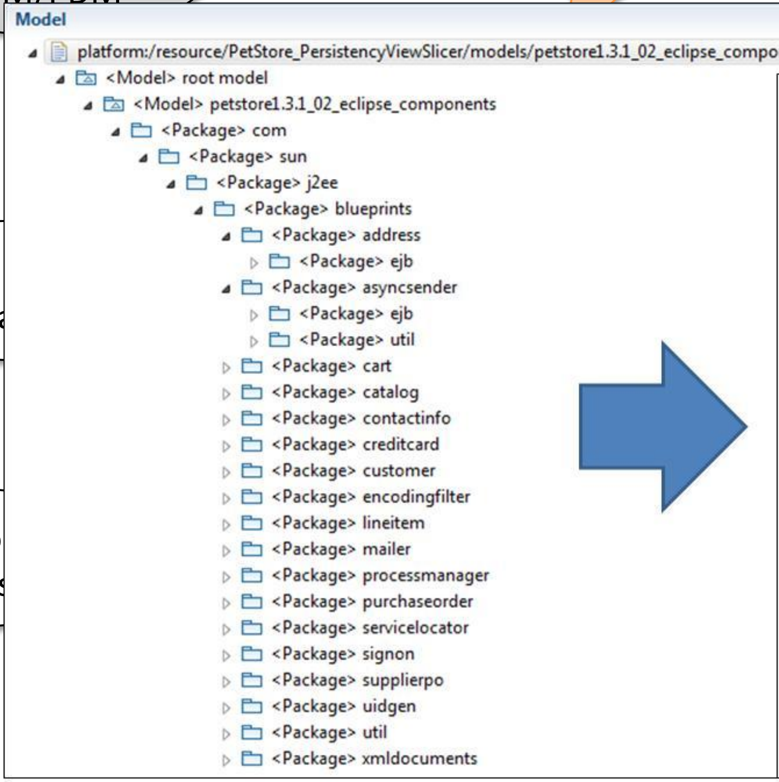
Model-Based Migration Approach in ARTIST

Slicing, Views, Queries, Tagging, …

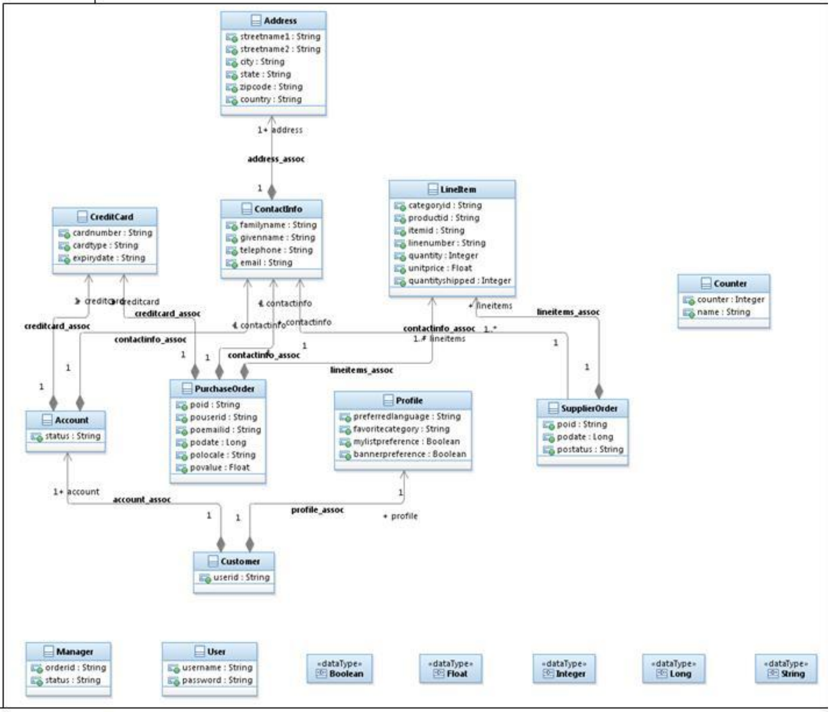
Separation PIM/PDM

UML Extra

Generatio Bridges



```
Model
◢ platform:/resource/PetStore_PersistencyViewSlicer/models/petstore1.3.1_02_eclipse_components.uml
  ◢ <Model> root model
    ◢ <Model> petstore1.3.1_02_eclipse_components
      ◢ <Package> com
        ◢ <Package> sun
          ◢ <Package> j2ee
            ◢ <Package> blueprints
              ◢ <Package> address
                ▷ <Package> ejb
              ◢ <Package> asyncsender
                ▷ <Package> ejb
                ▷ <Package> util
              ▷ <Package> cart
              ▷ <Package> catalog
              ▷ <Package> contactinfo
              ▷ <Package> creditcard
              ▷ <Package> customer
              ▷ <Package> encodingfilter
              ▷ <Package> lineitem
              ▷ <Package> mailer
              ▷ <Package> processmanager
              ▷ <Package> purchaseorder
              ▷ <Package> servicelocator
              ▷ <Package> signon
              ▷ <Package> supplierpo
              ▷ <Package> uidgen
              ▷ <Package> util
              ▷ <Package> xmldocuments
```



Model-Based Migration Approach in ARTIST

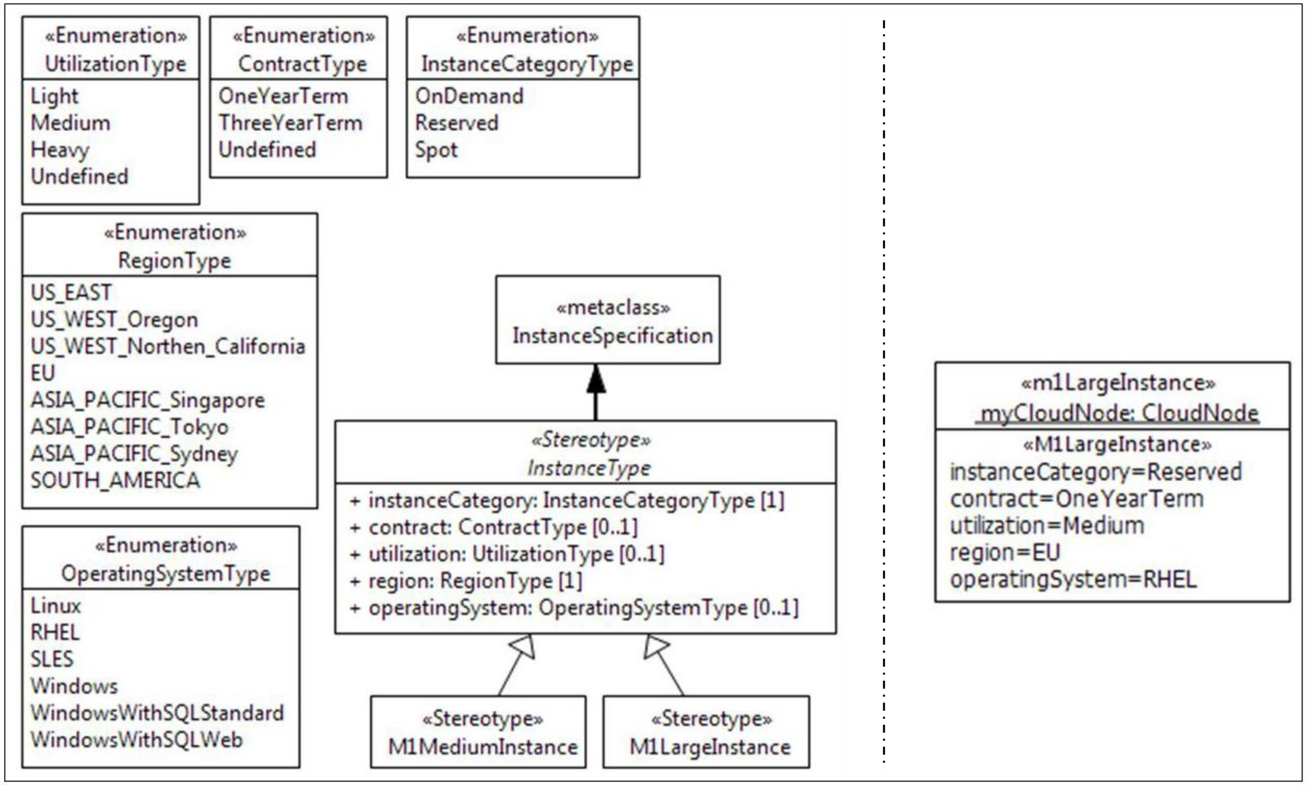
Separation PIM/PDM

Slicing, Views, Queries, Tagging, …

Modeling for the Cloud

UML M
Extract

Generation c
Bridges



«Enumeration»
UtilizationType
Light
Medium
Heavy
Undefined

«Enumeration»
ContractType
OneYearTerm
ThreeYearTerm
Undefined

«Enumeration»
InstanceCategoryType
OnDemand
Reserved
Spot

«Enumeration»
RegionType
US_EAST
US_WEST_Oregon
US_WEST_Northen_California
EU
ASIA_PACIFIC_Singapore
ASIA_PACIFIC_Tokyo
ASIA_PACIFIC_Sydney
SOUTH_AMERICA

«Enumeration»
OperatingSystemType
Linux
RHEL
SLES
Windows
WindowsWithSQLStandard
WindowsWithSQLWeb

«metaclass»
InstanceSpecification

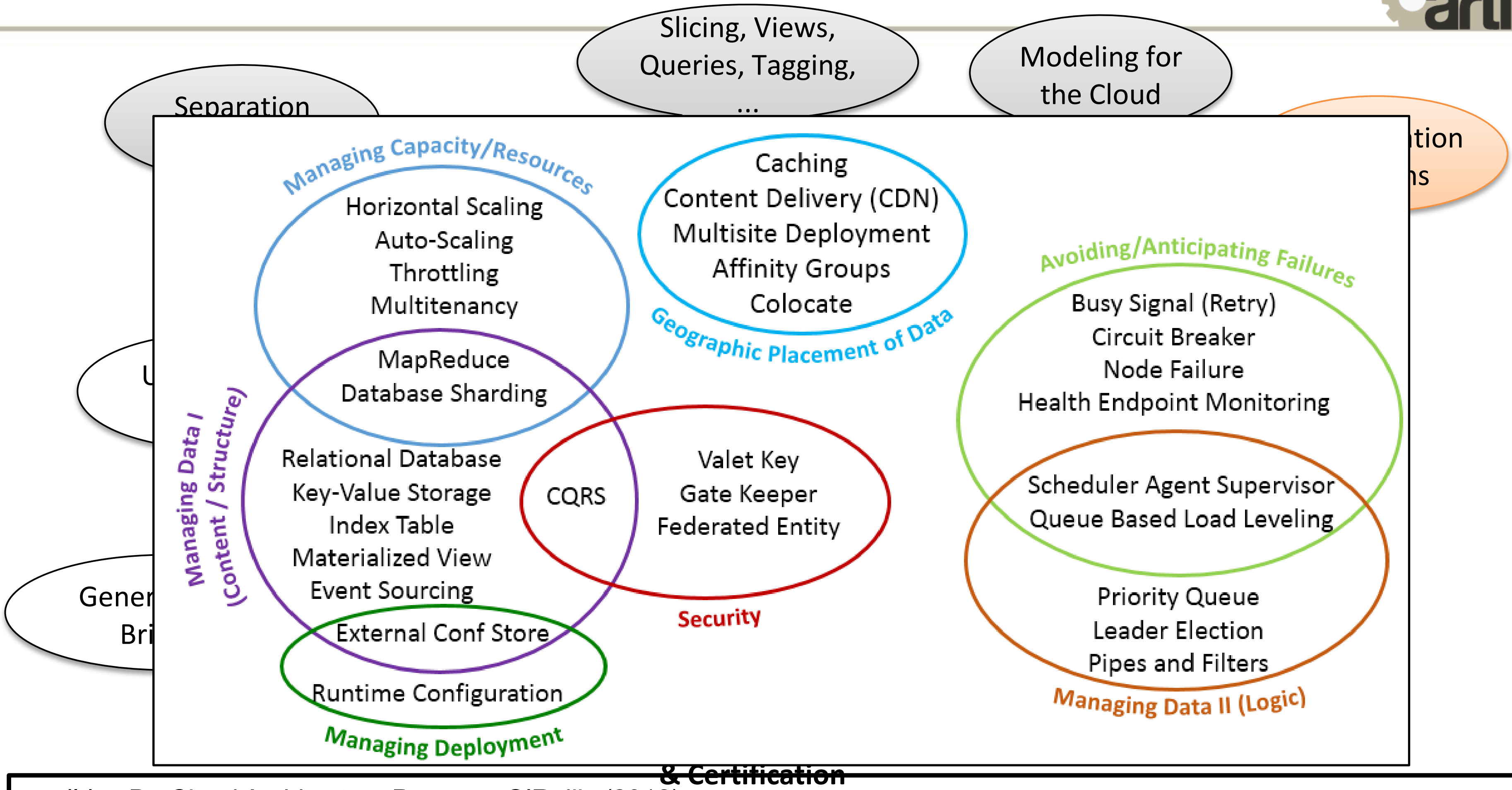
«Stereotype»
InstanceType
+ instanceCategory: InstanceCategoryType [1]
+ contract: ContractType [0..1]
+ utilization: UtilizationType [0..1]
+ region: RegionType [1]
+ operatingSystem: OperatingSystemType [0..1]

«Stereotype»
M1MediumInstance

«Stereotype»
M1LargeInstance

«m1LargeInstance»
myCloudNode: CloudNode
«M1LargeInstance»
instanceCategory=Reserved
contract=OneYearTerm
utilization=Medium
region=EU
operatingSystem=RHEL

Model-Based Migration Approach in ARTIST



- Wilder, B.: Cloud Architecture Patterns. O'Reilly (2012)

- Fehling, C., Leymann, F., Retter, R., Schupeck, W., Arbitter, P.: Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer (2014)

- Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1 edn. (1994)

- Homer, A., Sharp, J., Brader, L., Narumoto, M., T., S.: Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications. Microsoft Patterns & Practices (2014)

Model-Based Migration Approach in ARTIST

Separation PIM/PDM

Slicing, Views, Queries, Tagging, …

Modeling for the Cloud

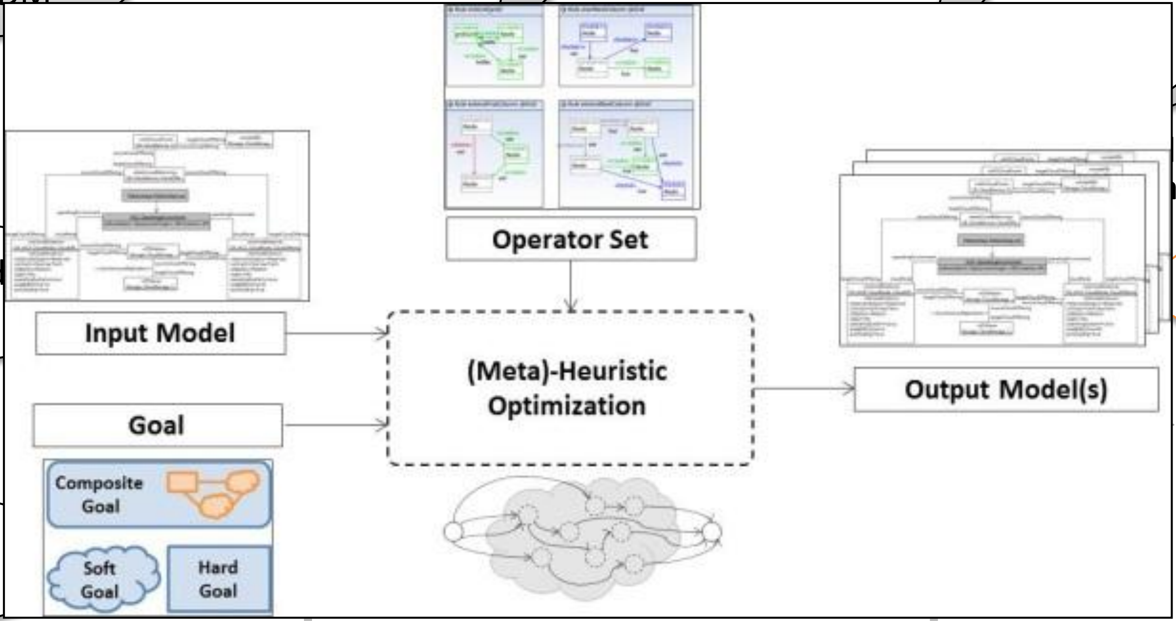
Cloudification Patterns

UML Mod Extractio

tion

Design Space Exploration

Generation of Bridges



Migration Validation & Certification

Model-Based Migration Approach in ARTIST

Slicing, Views,

Modeling for

Separation PIM/PDM

Cloudification Patterns

Mod
Understa

el
zation

UML Model Extraction

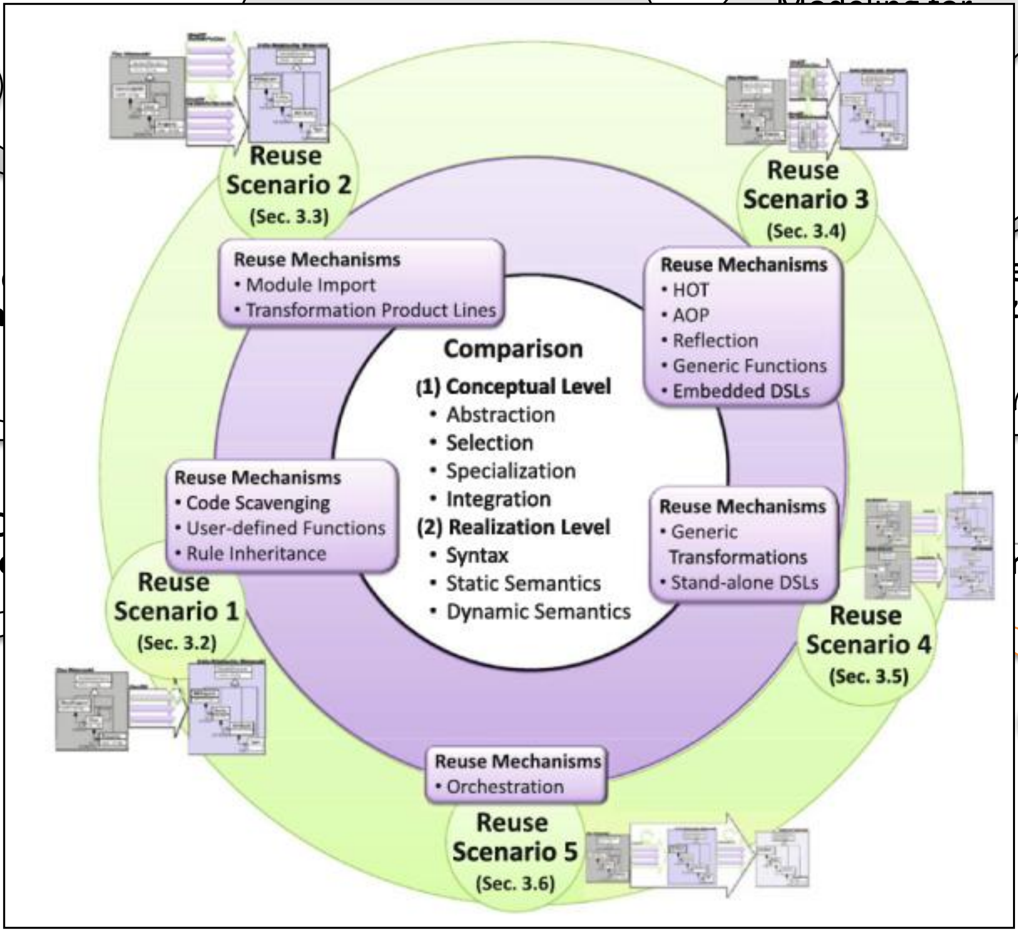
Design Space Exploration

C
Inj

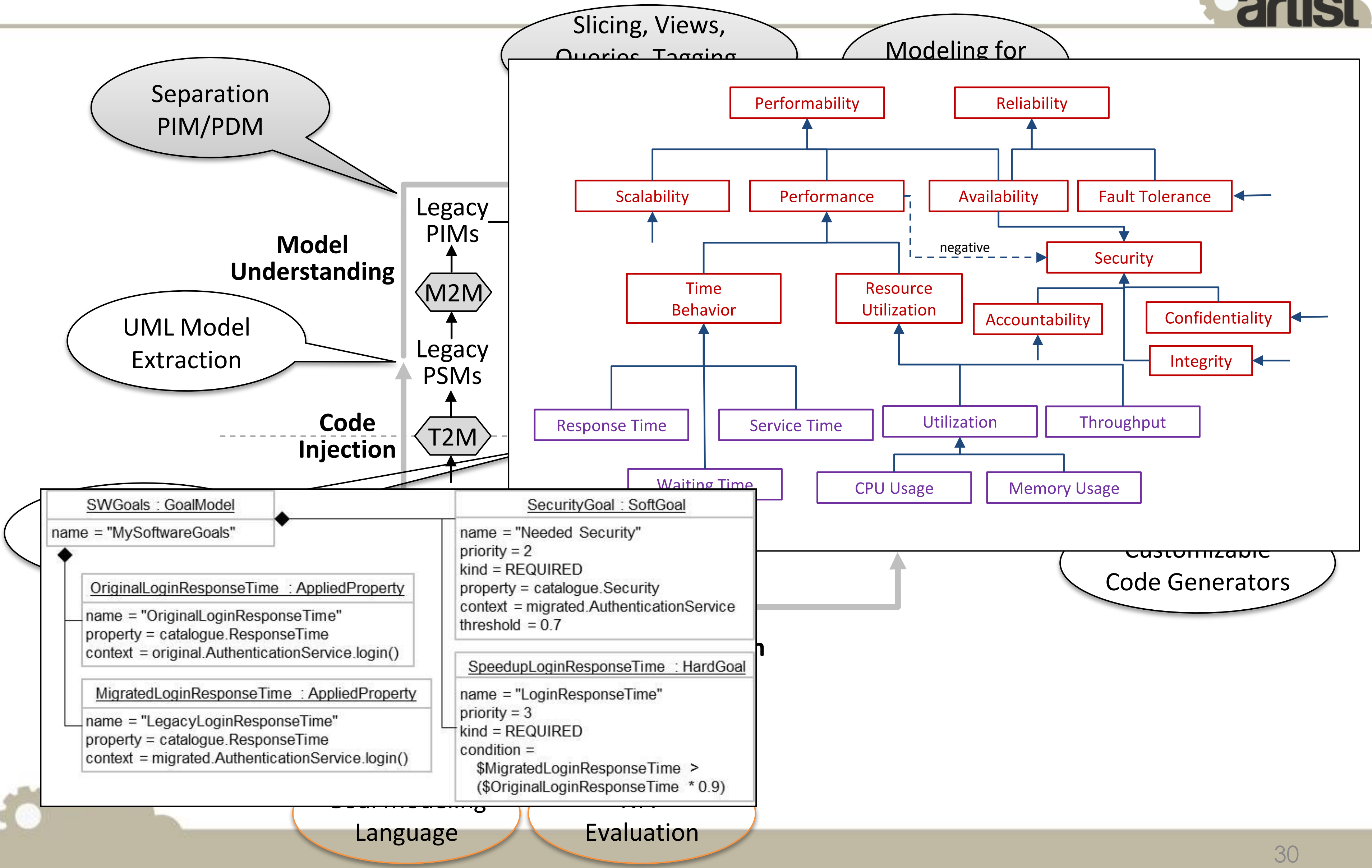
n

Generation of Bridges

Customizable Code Generators


Reuse Scenario 2 (Sec. 3.3)
Reuse Scenario 3 (Sec. 3.4)
Reuse Mechanisms • Module Import • Transformation Product Lines
Reuse Mechanisms • HOT • AOP • Reflection • Generic Functions • Embedded DSLs
Comparison
(1) Conceptual Level • Abstraction • Selection • Specialization • Integration
(2) Realization Level • Syntax • Static Semantics • Dynamic Semantics
Reuse Mechanisms • Code Scavenging • User-defined Functions • Rule Inheritance
Reuse Mechanisms • Generic Transformations • Stand-alone DSLs
Reuse Scenario 1 (Sec. 3.2)
Reuse Scenario 4 (Sec. 3.5)
Reuse Mechanisms • Orchestration
Reuse Scenario 5 (Sec. 3.6)

Model-Based Migration Approach in ARTIST



SWGoals : GoalModel
name = "MySoftwareGoals"

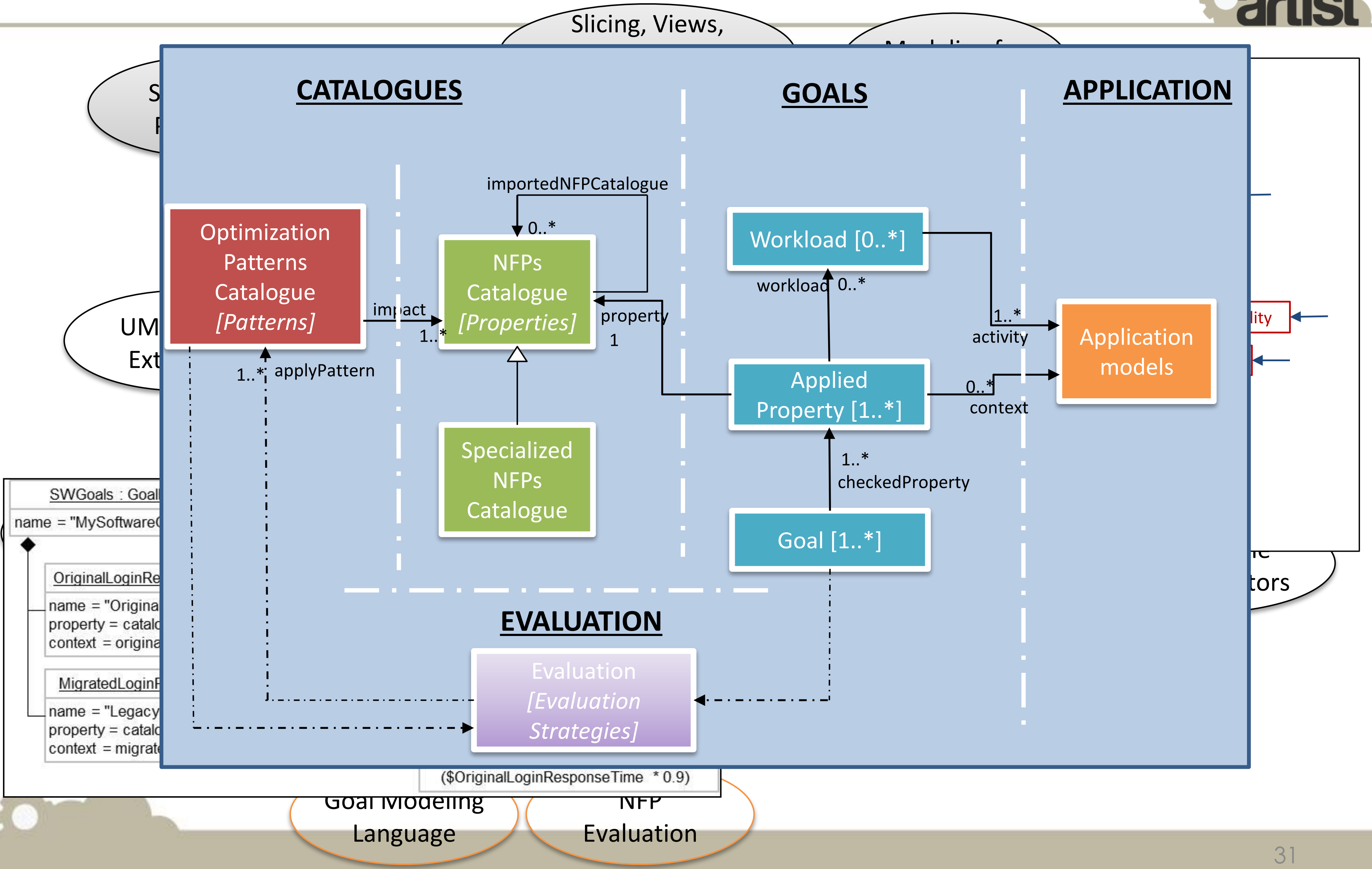
OriginalLoginResponseTime : AppliedProperty
name = "OriginalLoginResponseTime"
property = catalogue.ResponseTime
context = original.AuthenticationService.login()

MigratedLoginResponseTime : AppliedProperty
name = "LegacyLoginResponseTime"
property = catalogue.ResponseTime
context = migrated.AuthenticationService.login()

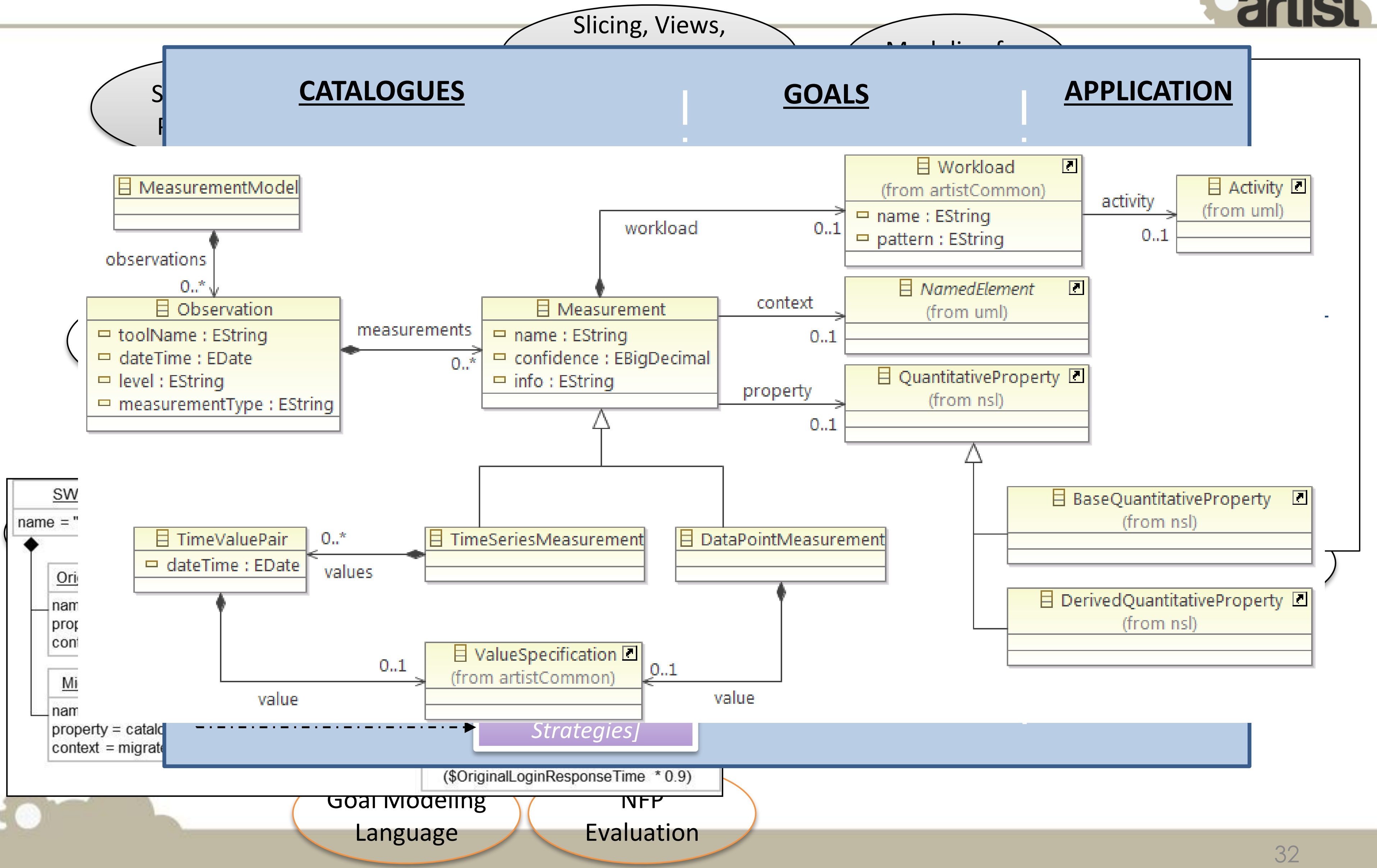
SecurityGoal : SoftGoal
name = "Needed Security"
priority = 2
kind = REQUIRED
property = catalogue.Security
context = migrated.AuthenticationService
threshold = 0.7

SpeedupLoginResponseTime : HardGoal
name = "LoginResponseTime"
priority = 3
kind = REQUIRED
condition =
  $MigratedLoginResponseTime >
  ($OriginalLoginResponseTime * 0.9)

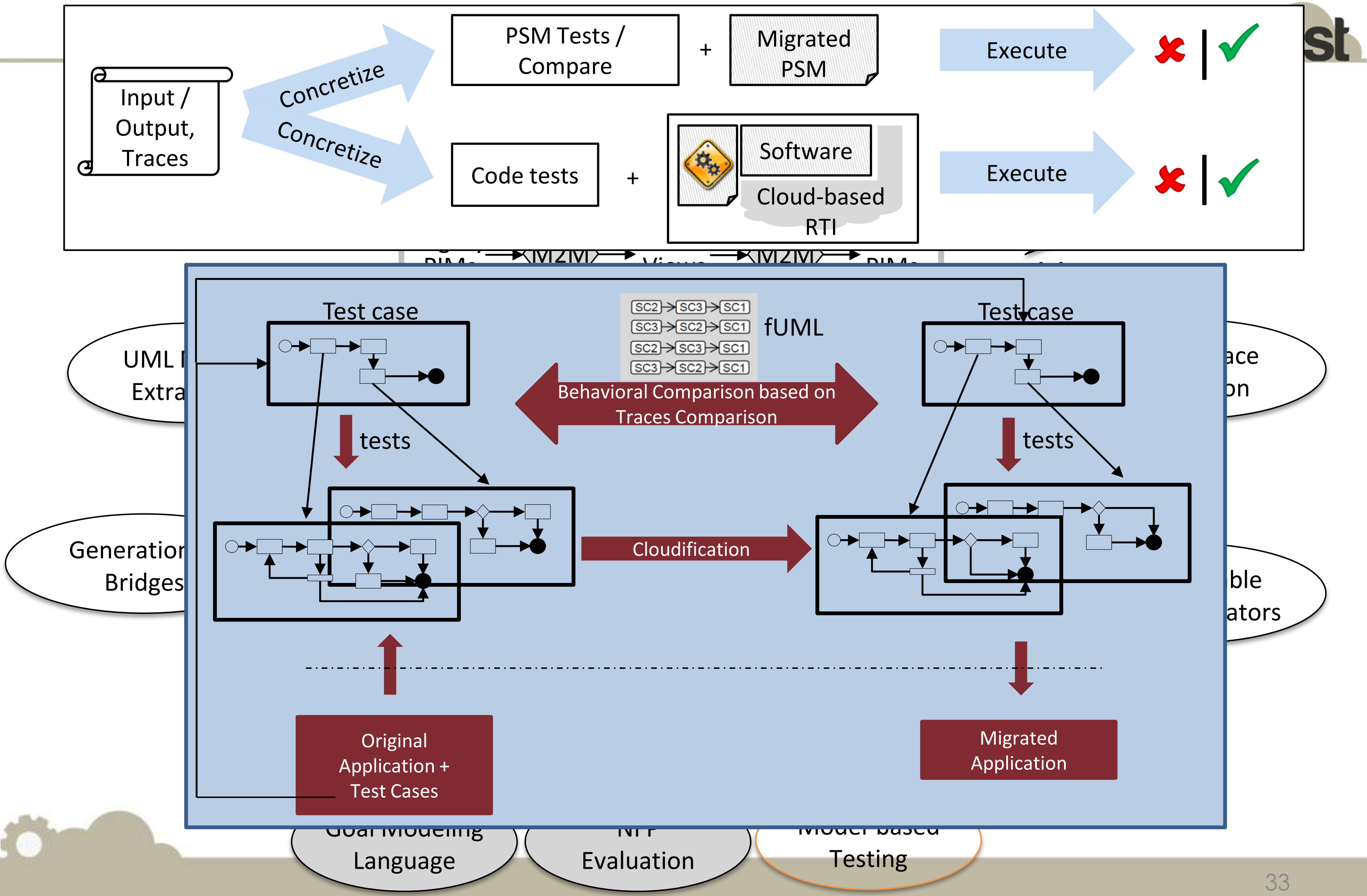
Model-Based Migration Approach in ARTIST



Model-Based Migration Approach in ARTIST



Model-Based Migration Approach in ARTIST



Model-Based Migration Approach in ARTIST

Slicing, Views,
Queries, Tagging,

Modeling for

Separat
PIM/PD

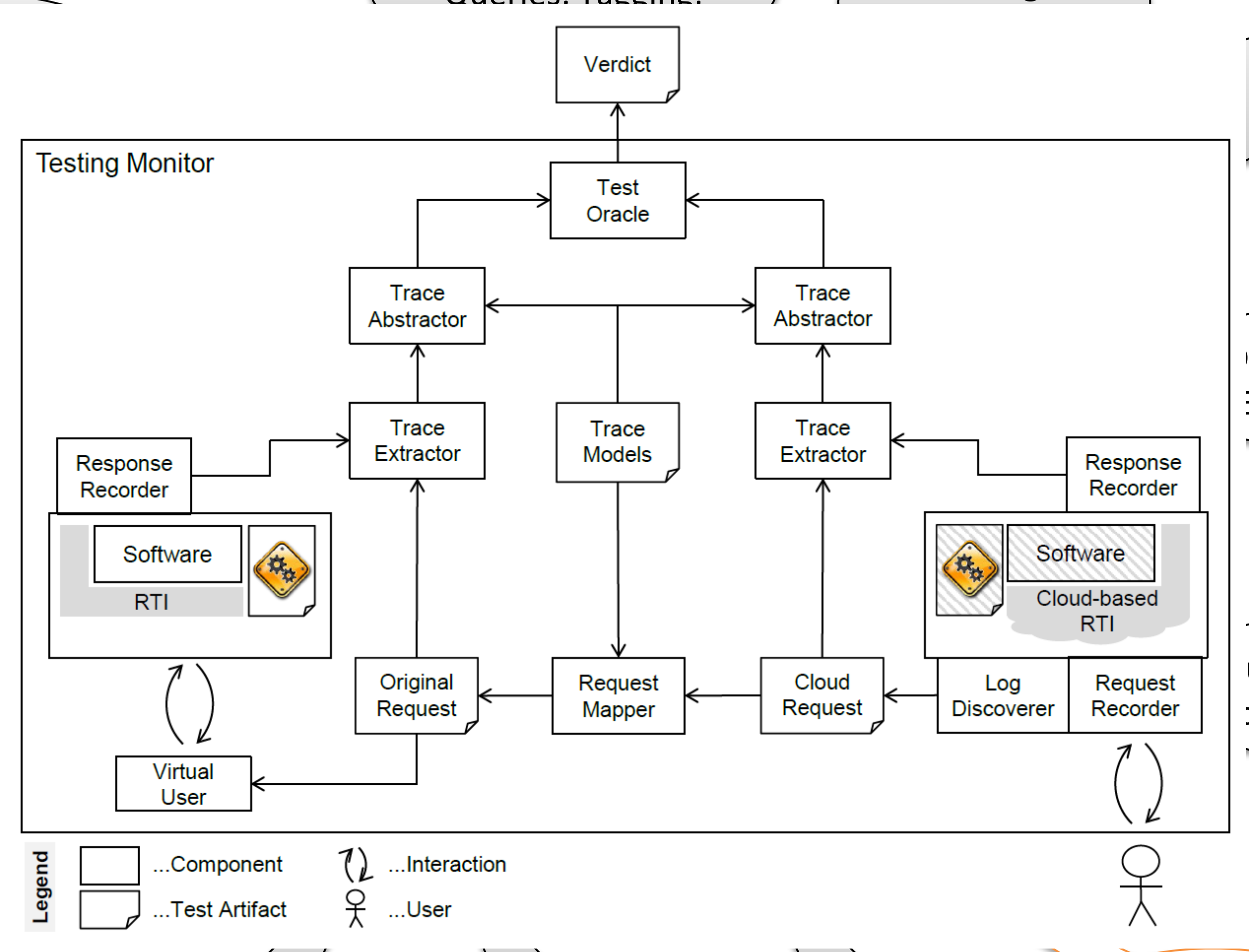
oudification
Patterns

UML Mod
Extractio

esign Space
Exploration

Generation of
Bridges

ustomizable
de Generators



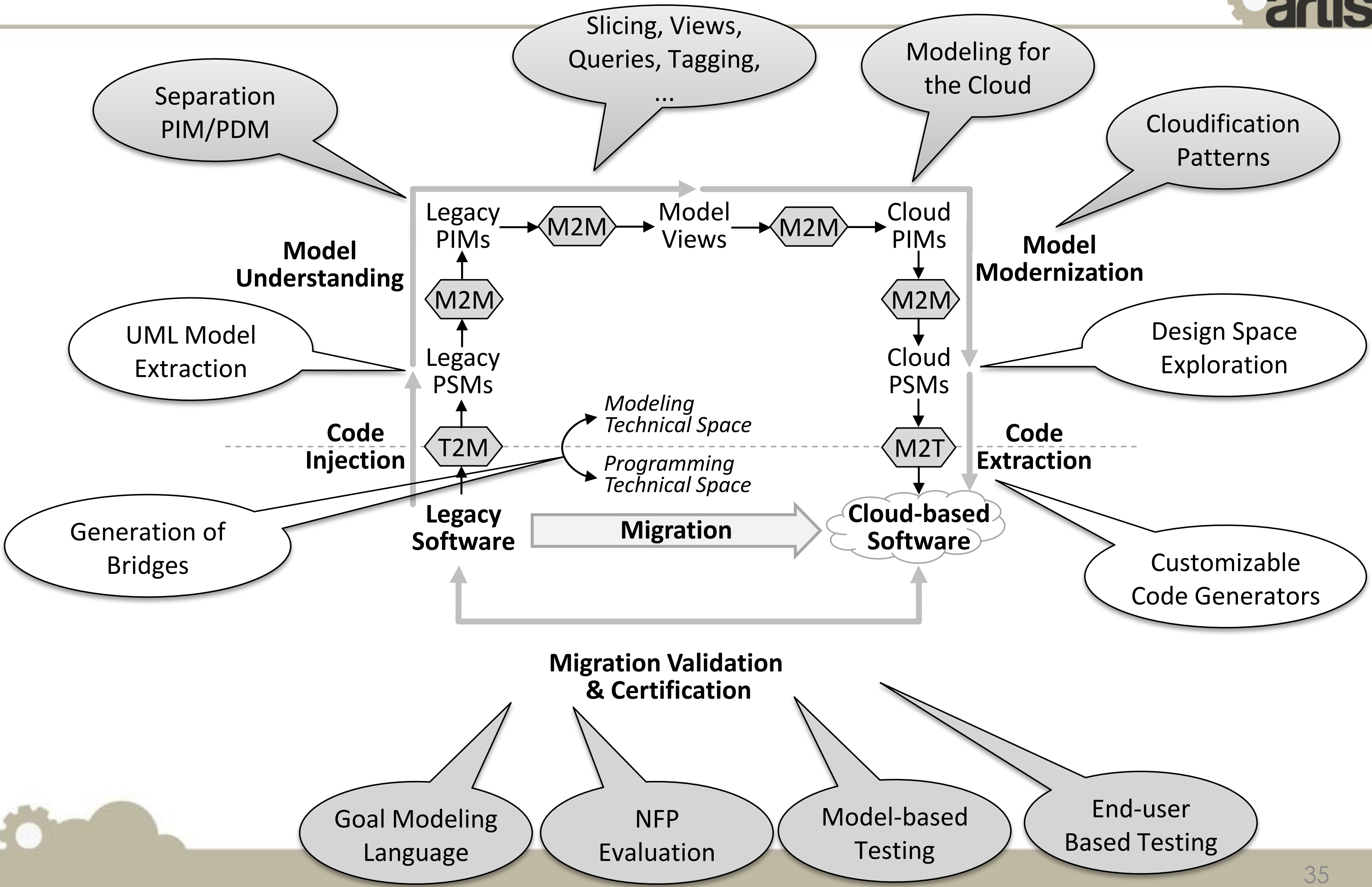
Goal Modeling
Language

NFP
Evaluation

Model-based
Testing

End-user
Based Testing

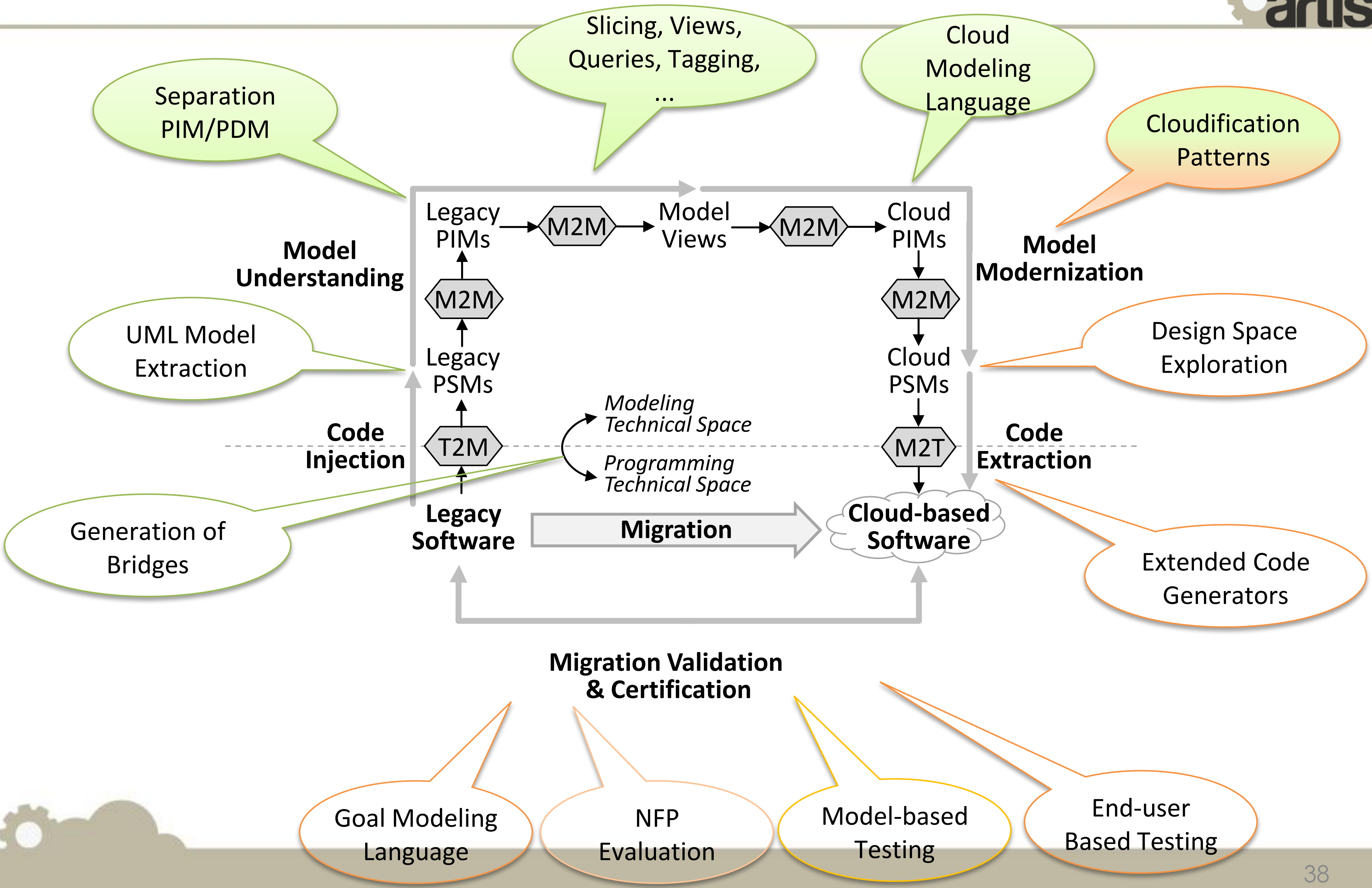
Model-Based Migration Approach in ARTIST



Questions and feedback

THANKS!

Current Status


Slicing, Views, Queries, Tagging, …
Cloud Modeling Language
Separation PIM/PDM
Cloudification Patterns
Legacy PIMs
M2M
Model Views
M2M
Cloud PIMs
Model Understanding
M2M
M2M
Model Modernization
UML Model Extraction
Legacy PSMs
Cloud PSMs
Design Space Exploration
Modeling Technical Space
Code Injection
T2M
M2T
Code Extraction
Programming Technical Space
Generation of Bridges
Legacy Software
Migration
Cloud-based Software
Extended Code Generators
Migration Validation & Certification
Goal Modeling Language
NFP Evaluation
Model-based Testing
End-user Based Testing