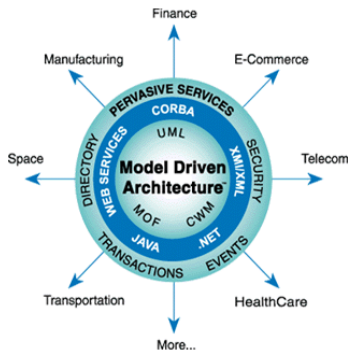
Model Driven Architecture :

principles and practice

Report about paper “Model driven architecture: Principles and practice” by Alan Brown

Reporter: Tian Zhang

Nanjing University



About the paper and journal

- Software and System Modeling (SoSyM)
  - Springer Berlin Heidelberg
  - quarterly international journal
  - emphasis on :
    - theoretical foundations of *modeling languages* and *techniques*
    - rigorous analyses of "real-world" *modeling experiences*
- Model driven architecture: Principles and practice
  - Published online: 3 August 2004 - *Springer-Verlag 2004*
  - 14 pages, 2 parts, 6 sections

About the author

- Alan William Brown *(MDA thought leaders at IBM)*
  - 88: University of Newcastle upon Tyne, Ph.D
  - 88-91: Lecturer in University of York
  - 91-96: SEI, Carnegie Mellon University
    - CASE Environments project
  - 96-present: Texas Instruments, Computer Associates, Catapulse, (2001)IBM Rational software
  - http://www-306.ibm.com/software/rational/bios/brown.html
  - http://www.jorvik.com/alanbrown/

Software difficulties

- Difficulties:
  - understanding highly complex business domains
  - large teams of engineers
  - over multiple phases of a project spanning many months
  - time-to-market pressures
  - complicated assortment of infrastructure technologies
  - variety of middleware acquired from many vendors
  - … …
- There are two important ideas:
  - Service-Oriented Architectures
  - Software Product Lines

Service oriented architectures

- Informal definition
  - viewing enterprise solutions as federations of services connected via well-specified contracts that define their service interfaces
- Most notable example
  - Web Services – when the services are distributed across multiple machines and connected by the Internet.

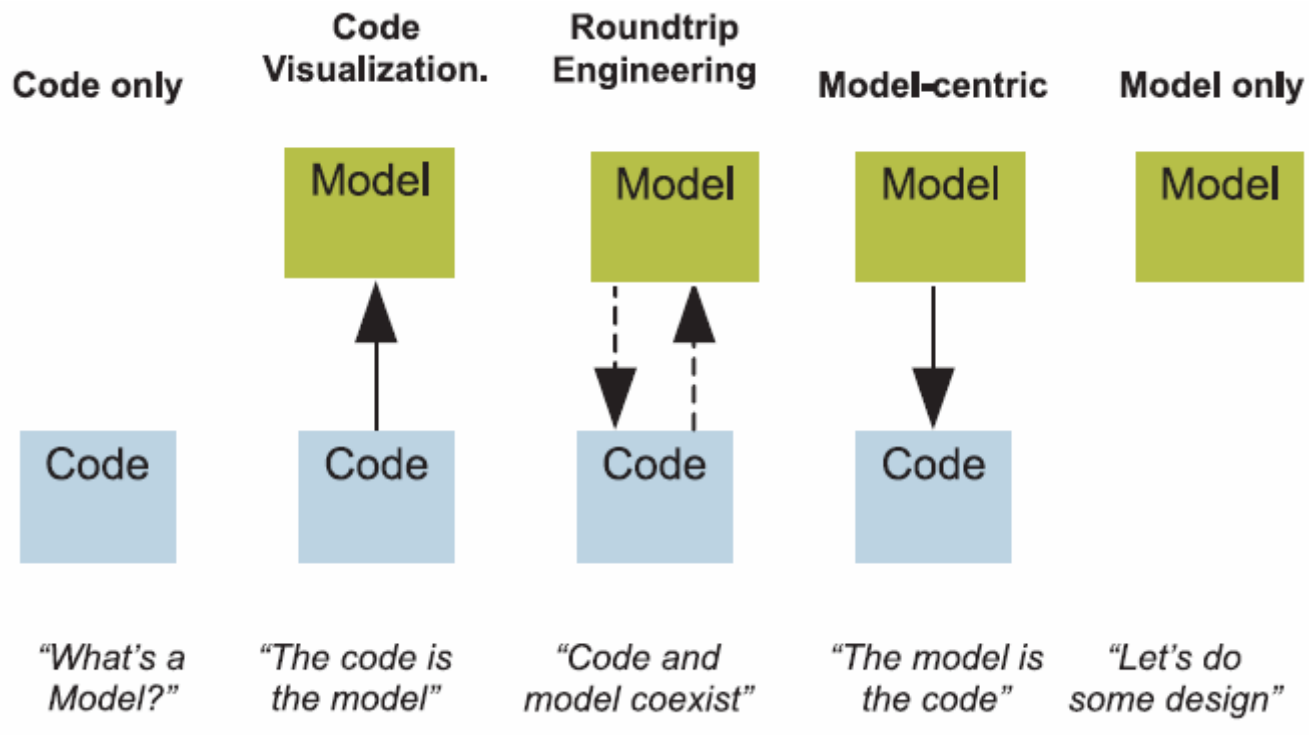
Software product lines

- Informal definition
  - referring to engineering techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production
- More generally
  - the underpinning of any product line approach can be viewed as a way to transform *descriptions* of a solution at one level of abstraction into *descriptions* at a lower level of abstraction by applying well-defined patterns.

The new development imperative

- The need for greater flexibility in the development of enterprise-scale solutions
- OMG has created a conceptual framework to
  - separate business-oriented decisions from platform decision
  - allow greater flexibility when architecting and evolving systems
- Conceptual framework is a set of key concepts and structures
  - Unified Modeling Language (UML)
  - Meta-Object Facility (MOF)
  - Common Warehouse Meta-model (CWM)
  - … …
- OMG calls this framework Model Driven Architecture

Review current practice of modeling approach



- Figure illustrates the spectrum of modeling approaches in use by software practitioners today.

Code only approach

- Relying almost entirely on the code
- Using no separately defined models at all
- Expressing models of the system directly in 3rdGL within an IDE
  - 3rdGL: Java, C++ and C#
  - IDE: Rational Software Architect and Microsoft Visual Studio
- Difficulties
  - hard to understand the key characteristics of the system
  - hard to manage the evolution of these solutions as
    - their scale and complexity increases
    - the system evolves over time
    - the original members of the team leave

Code visualization approach

- Visualizing the code through some graphical notation
- It's possible to manipulate graphical notations as an alternative to editing source code
  - visual rendering becomes a direct representation of the source code
  - such rendering is sometimes called a code model, or an implementation model, or a diagram
  - "model" is reserved for higher level abstraction
- Limitations
  - the diagrams are tightly coupled representations of the code
  - the abstraction of diagrams are at the code level

Roundtrip engineering (RTE) approach

- RTE is between the abstract models and the code
  - an abstract model of the system describing the system architecture of design
- For instance
  - design team works on the high level provides design models
  - implementation team converts them into code implementation
- Limitations
  - typically the tools generate code stubs only
  - errors and their corrections are made in the both levels
  - changes made to the code must be reconciled with the original models (hence the term “roundtrip engineering”)

Definition of **RTE** from some papers

- **Definition 1** (Simple definition of Round-trip Engineering)
  - Let D be the design of the product P, let r be a reverse engineering procedure such that r(P) = D and let g be a product generation procedure such that g(D) = P. Iff g(r(P)) P (and thus r(g(D)) D) then hg, ri is a round-trip engineering system.

[A Definition of Round-trip Engineering, 2004]

- **Definition 2** (informal definition)
  - the seamless integration between design diagrams and source code, between modeling and implementation.
  - *generating* code from a design diagram, *changing* that code in a separate development environment and *recreating* the adapted design diagram back from the source code.

[UML Shortcomings for Coping with Round-trip Engineering, 1999]

关于文中Roundtrip Engineering的理解

- Alan Brown的定义和其他人的定义有些不同
  - Alan没有强调model和code间的无缝衔接；
  - Alan认为工具还只能简化model->code的转换，生成code stubs；
  - Alan在文中强调的是design models和implementation的分离；
  - 事实上Alan也并没有针对不同分类给出严格定义；
  - Alan只是认同roundtrip engineering包含了forward和reverse两个过程；
- 可以看出Alan在文中介绍的建模分类情况应该是一种工业界现有的实际情况，而不是学术研究层面的术语

Model-centric approach

- In this approach, models are established in sufficient detail that the full implementation of the system can be generated from the models themselves.
- To achieve this, the models may include
  - representations of the persistent and non-persistent data
  - business logic
  - presentation elements
  - interfaces to data and services
- Typically requires
  - standard or proprietary application frameworks
  - runtime services
  - tools supporting generation of particular styles of applications

Model only approach

- Developers use models purely as thought aids
  - in understanding the business or solution domain
  - for analyzing the architecture of a proposed solution
  - for discussion, communication and analysis among teams
  - in proposals for new work
  - for establishing a shared vocabulary and set of concepts
- Implementation may be practically disconnected from models
  - Example: to **outsource** implementation and maintenance of the system while maintaining control of the overall architecture

From models to MDA

- Interesting questions
  - Which of these approaches can we describe as “model-driven”?
  - If I create a visualization of some part of a system, does that mean I am practicing MDA?
- Answer
  - Unfortunately, there is no definitive answer.
- Alan的这篇文章是在2004年8月发表的，在同年8月OMG的ORMSC (Object and Reference Model Subcommittee)全体代表大会上给出了MDA的原则性定义。
  - MDA Guide v1.0.1中没有给出MDA的明确定义；
  - OMG站点上也没有以Specification的形式给出MDA的定义；
  - 也许这就是本文名为“principles and practice”的原因。

ORMSC's Principle Definition

The following was approved unanimously by 17 participants at the ORMSC plenary session, meeting in Montreal on 23 August 26, 2004. The stated purpose of these two paragraphs was to provide principles to be followed in the revision of the MDA Guide.

> MDA is an OMG initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. Not all of these technologies will directly concern the transformations involved in MDA.
>
> MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modeling language must be used. Any modeling language used in MDA must be described in terms of the MOF language, to enable the metadata to be understood in a standard manner, which is a precondition for any ability to perform automated transformations.

MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modeling language must be used. Any modeling language used in MDA must be described in terms of the MOF language, to enable the metadata to be understood in a standard manner, which is a precondition for any ability to perform automated transformations.

The principles of MDA

- Four principles that underlie the OMG's view of MDA
  - Models expressed in a well-defined notation are a cornerstone to understanding systems for enterprise-scale solutions *[UML]*
  - The building of systems can be organized around a set of models by imposing a series of transformations between models, organized into an architectural framework of layers and transformations *[PIM,PSM, QVT]*
  - A formal underpinning for describing models in a set of metamodels facilitates meaningful integration and transformation among models, and is the basis for automation through tools *[MOF, QVT]*
  - Acceptance and broad adoption of this model-based approach requires industry standards to provide openness to consumers, and foster competition among vendors

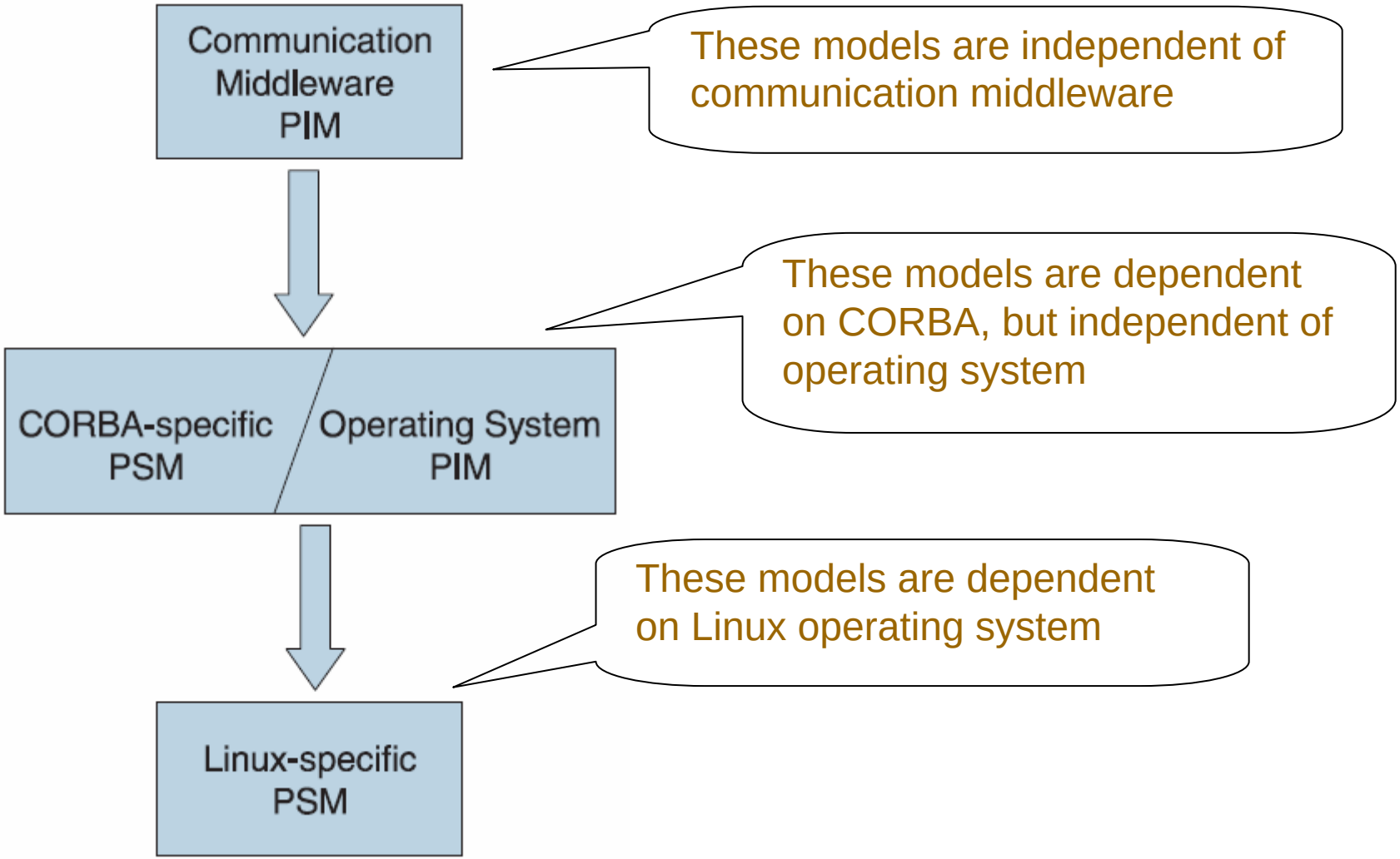
Concepts supporting these principles

- *MDA Viewpoints*
  - Computation Independent Viewpoint
  - Platform Independent Viewpoint
  - Platform Specific Viewpoint
- Four types of models
  - Computation Independent Model (CIM)
  - Platform Independent Model (PIM)
  - Platform Specific Model (PSM)
  - Platform Model
- *Implementation Specific Model (ISM) - ?*
  - Alan在文中提到这个ISM的概念并指出引自[MDA Guide v1.0.1]，遗憾的是我并未在该specification中找到这个概念

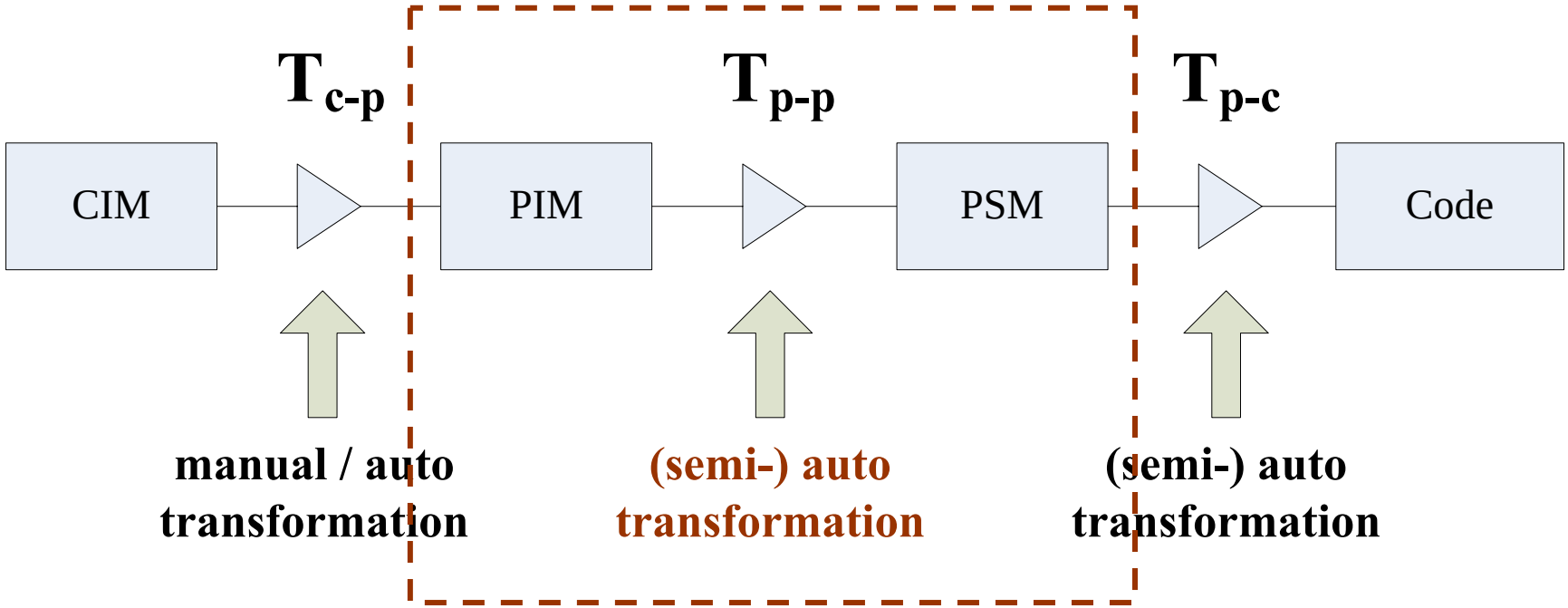
Note that

- Computation Independent Viewpoint
  - **focuses on** the on the environment of the system, and the requirements for the system;
  - **hide** the details of the structure and processing of the system
- Computation Independent Model (CIM)
  - CIM就是从CIV这样的视点去看这个系统时，所能看到的模型
  - A CIM is sometimes called a domain model or a business model
  - the importance of CIM
    - 在领域专家和系统设计专家之间架起一座桥梁
- Platform Model
  - 描述一个platform的组成部分以及platform上所提供的services
  - 描述应用该platform所要涉及的元素

PIM/PSM is not absolute



How is MDA Used? – OMG *[MDA Guide v1.0.1]*



Camps of MDA

- Suggested by Steve Cook, supported by Martin Flower
  - The UML PIM camp
    - MDA involves the use of UML to build Platform Independent Models (PIMs) which are transformed into Platform Specific Models (PSMs) from which code is generated.
  - The MOF camp
    - MDA does not involve the use of UML, but instead the crucial technology is MOF, and the definition of modelling languages and language transformations using MOF.
  - The Executable UML camp
    - MDA involves building a UML compiler, making it a first class programming language.

OMG's suggestion of $\mathbf{T}_{p\text{-}p}$ approaches -1

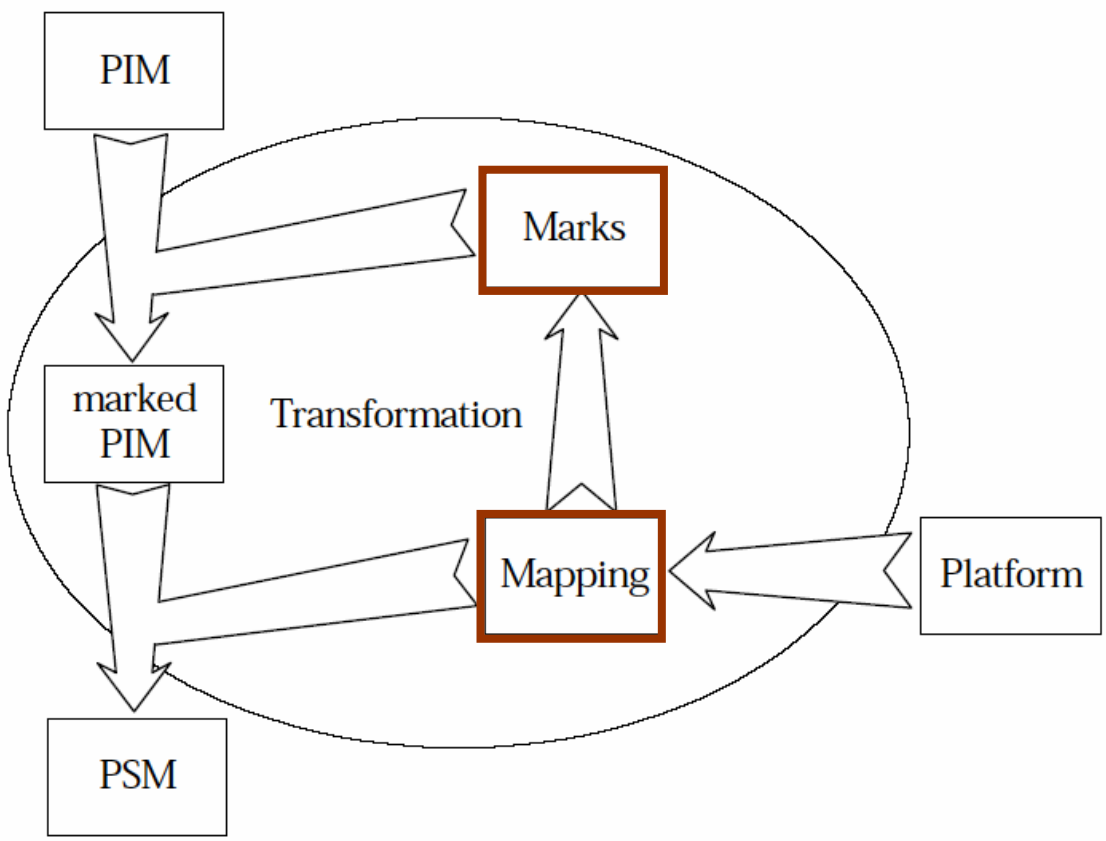


Figure3-1 Marking a Model

Mapping

- An MDA mapping provides specifications for transformation of a PIM into a PSM for a particular platform.
- *Example 1:*
  - *An EDOC ECA PIM contains attributes which indicate whether an Entity in that model is managed or not, and whether it is remote or not. A mapping from ECA to EJB will state that every managed ECA entity will result in a Home class, and that every remotable ECA entity will result in a RemoteInterface.*
- *Example 2:*
  - *A UML PIM to EJB mapping provides marks to be used to guide the PIM to PSM transformation. Marking a UML class with the Session mark resultsin the transformation of that class according to the mapping into a session beanand other supporting classes.*

Mapping sorts and facilities

- Mapping sorts
  - Model Type Mappings
  - Metamodel Mappings *(a specific example of MTM)*
  - Model Instance Mappings
  - Combined Type and Instance Mappings
- Mapping facilities
  - Marks
  - Templates
    - templates are parameterized models and like design patterns
  - Mapping Language

- Marks
  - Marks主要是针对model instance mappings而言的，可以被看作是应用到PIM模型上的一个“透明层”中
  - Marks可以有不同的来源，包括：
    - types from a model, specified by classes, associations, or other model elements
    - roles from a model, for example, from patterns
    - stereotypes from a UML profile
    - elements from a MOF model
    - model elements specified by any metamodel

OMG's suggestion of $T_{p\text{-}p}$ approaches -2

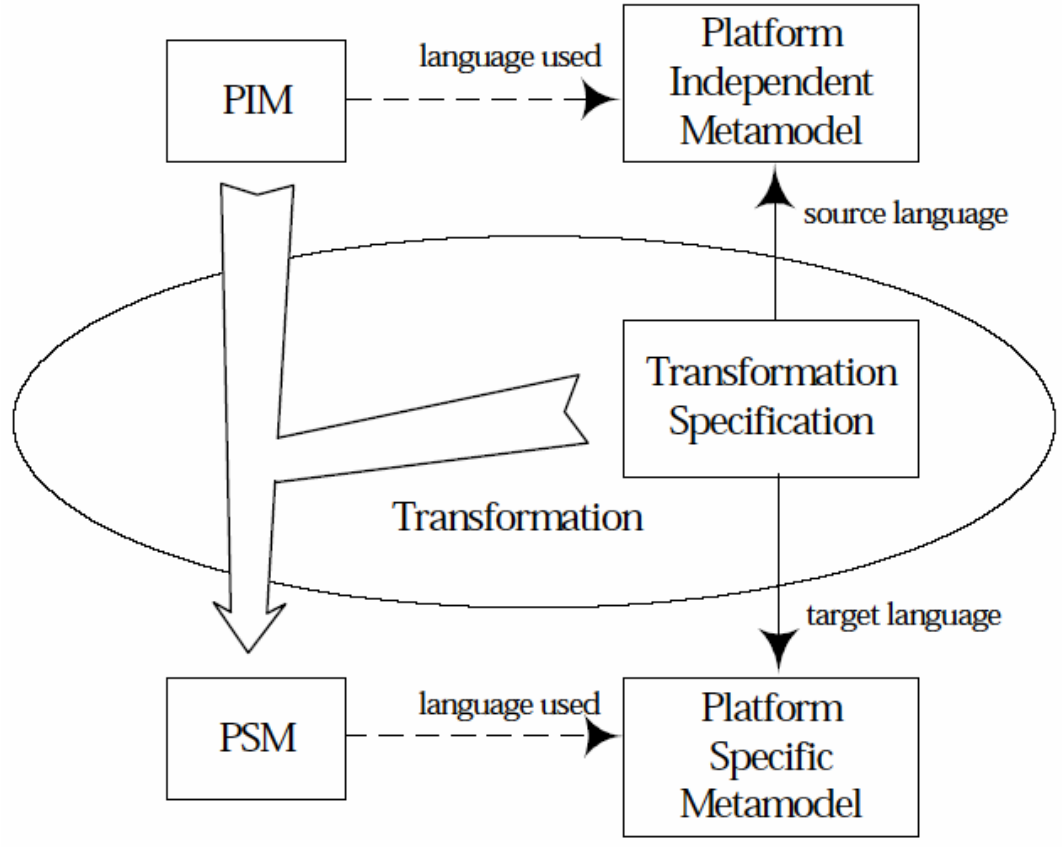
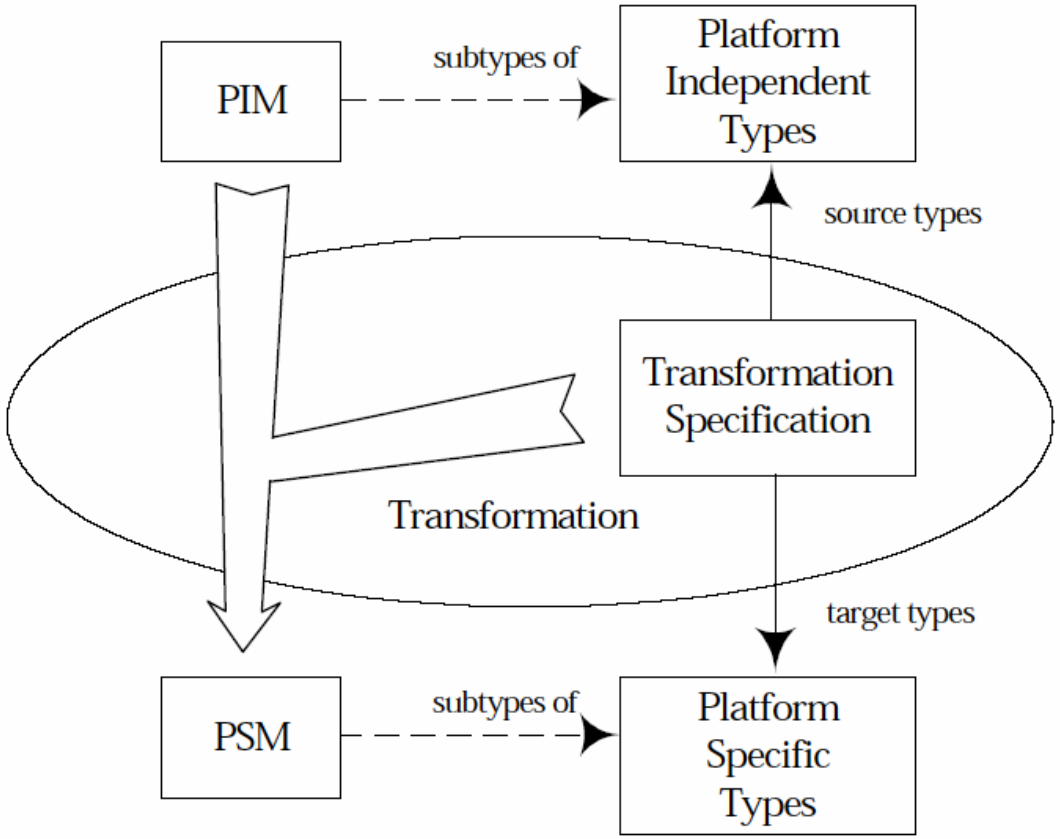


Figure3-2 Metamodel Transformation

Related work:

1. *[meta-02-1]*
2. *[meta-02-2]*

OMG's suggestion of $T_{p\text{-}p}$ approaches -3



Related work:

1. *[mt-03-1]*
2. *[mt-03-2]*

Figure3-3 Model Transformation

OMG's suggestion of $\mathbf{T}_{p\text{-}p}$ approaches -4

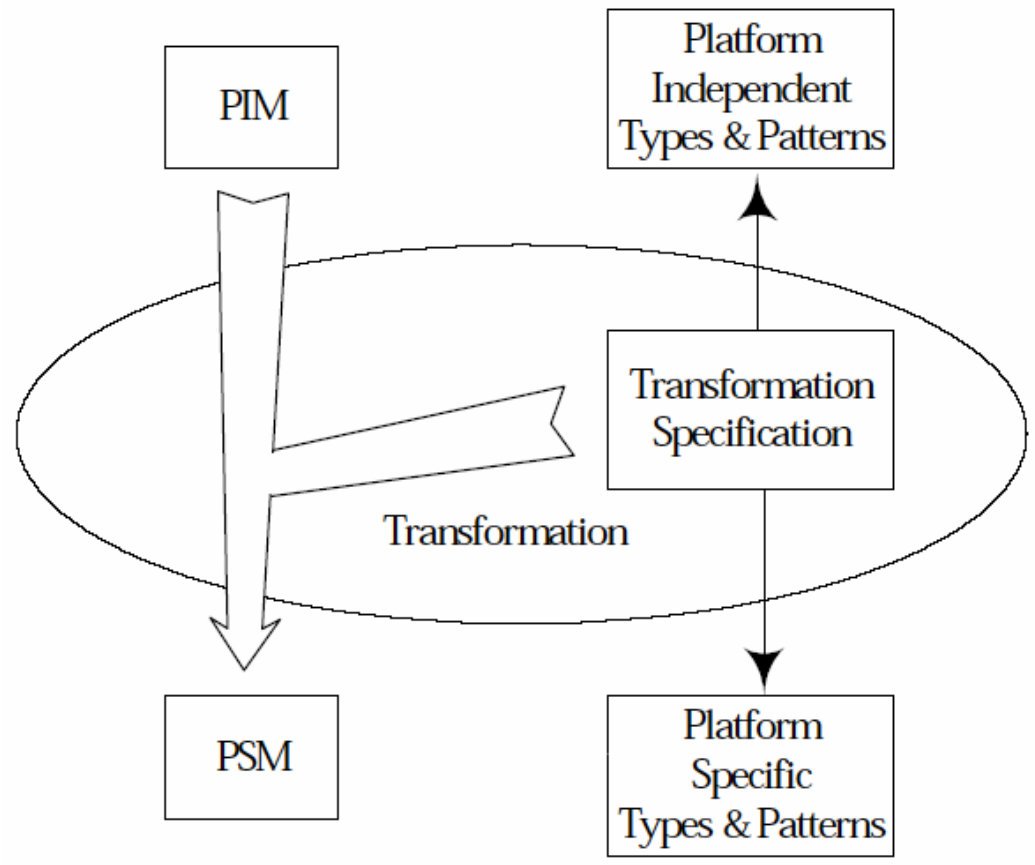


Figure3-4 Pattern Application

OMG's suggestion of $\mathbf{T}_{p\text{-}p}$ approaches -5

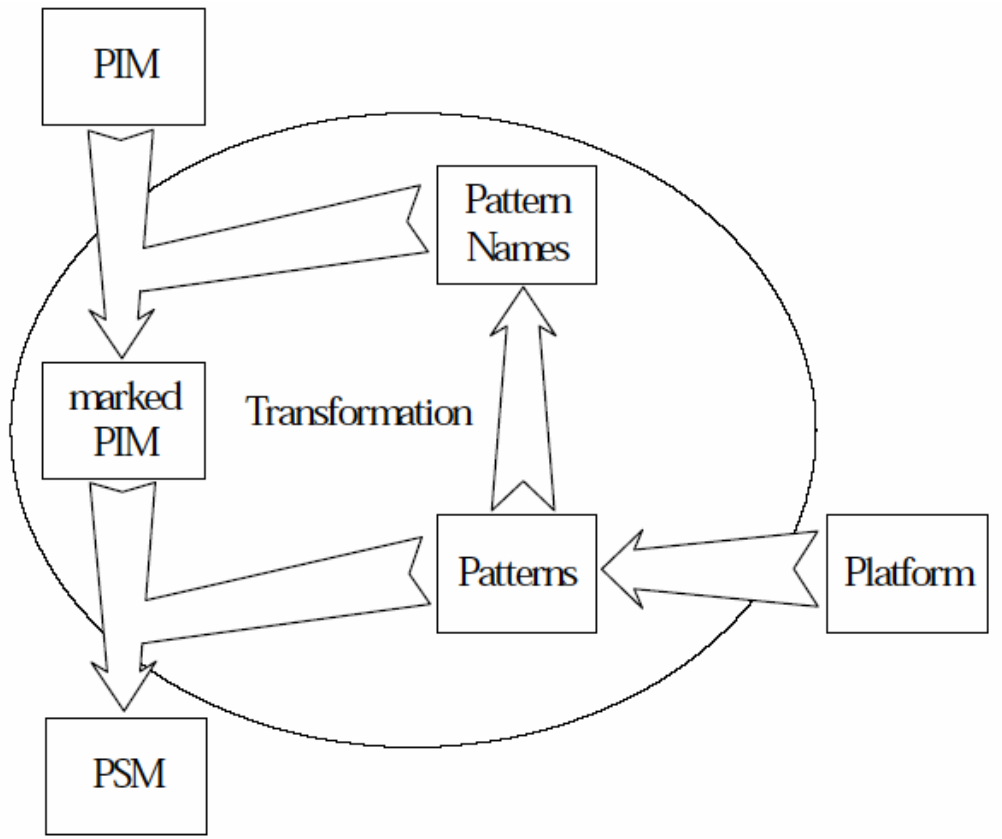


Figure3-5 Another way to use Patterns

Transformation (PIM-PSM)

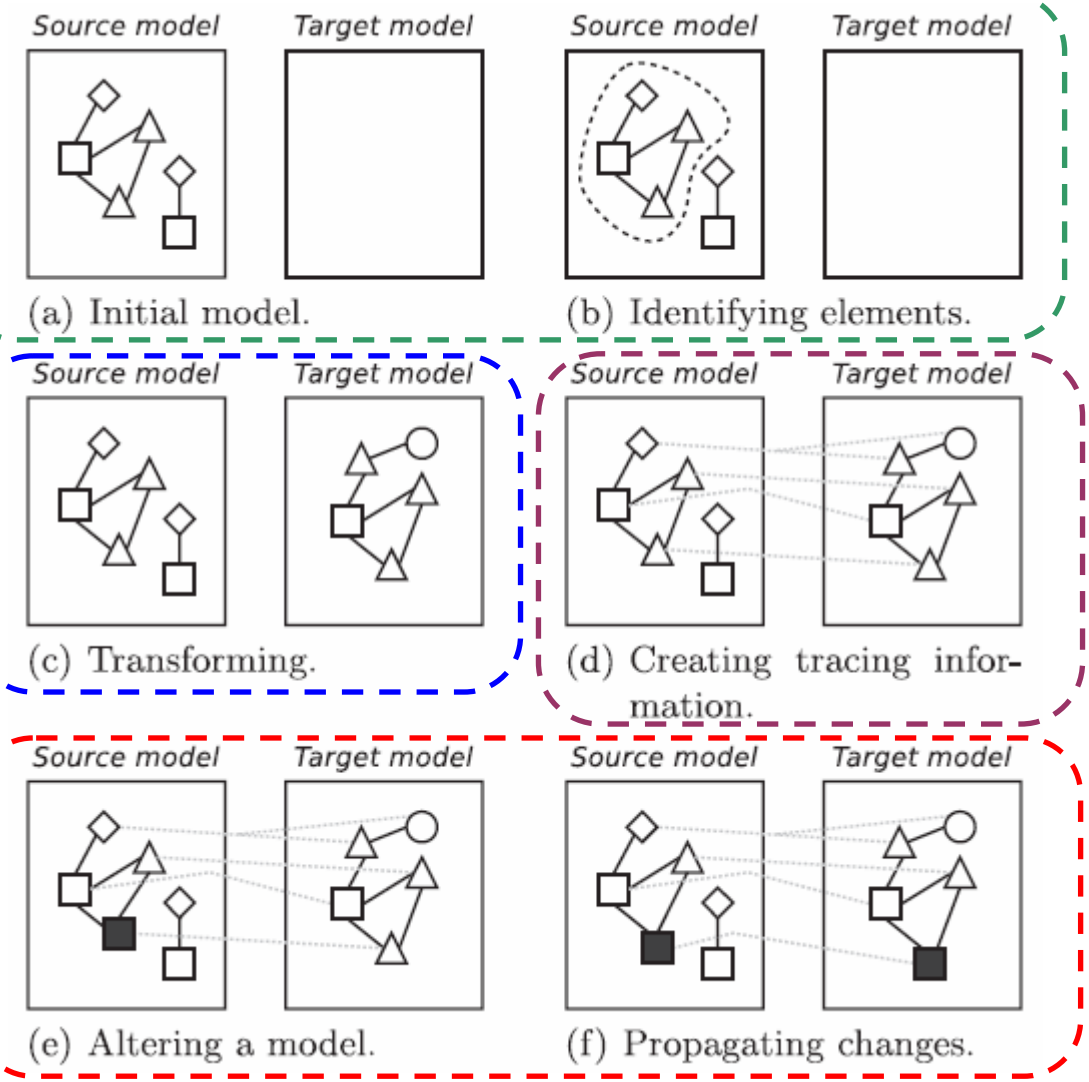
- Model transformation is the process of converting one model to another model of the same system.
  - The input to the transformation is the marked PIM and the mapping.
  - The result is the PSM and the record of transformation.
- Using model type mapping
  - transformation takes any PIM specified using one model and, following the mapping, produces a PSM specified using another model
- Using model instance mapping
  - transformation takes a marked PIM and, following the mappings, as indicated by the marks, produces a PSM.

Kinds of Input to Transformation

- Patterns
  - Patterns are also important in the description of groups of concepts in one model that correspond to a concept, or different group of concepts in another model when specifying a type-based transformation.
- Technical Choices
- Quality Requirements
  - A whole range of quality of service requirements can be used to guide transformations.
  - [*qos-04-1*]

Review and summary of the approaches above

Search and identify appropriate elements



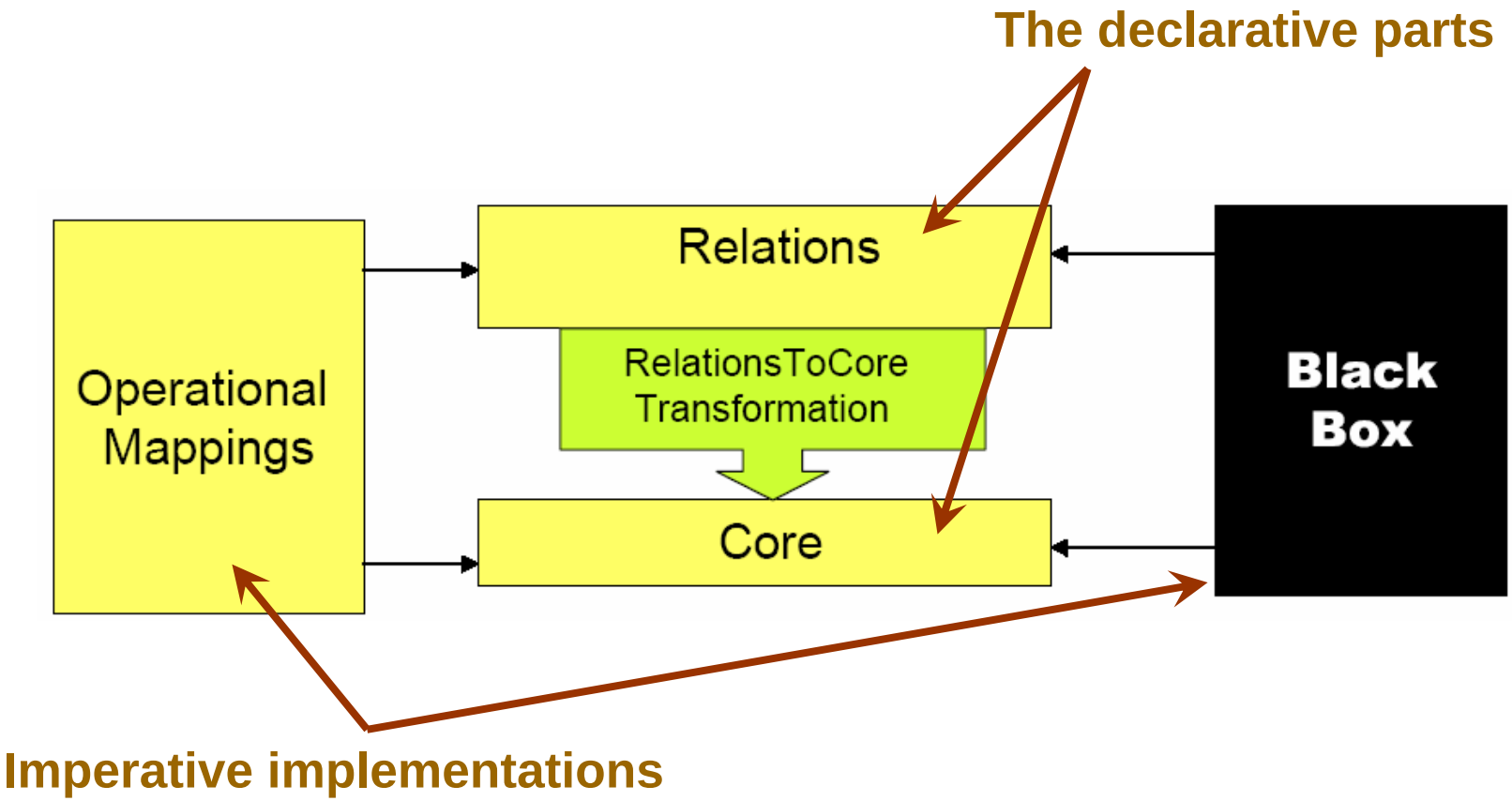
Retain the racing information

Detect and update models

MOF 2.0 Query/Views/Transformation (QVT)

- QVT is an OMG initiative aimed at creating a standard approach to model transformations.
  - On April, 24, 2002, the OMG issued a Request for Proposals (RFP) for MOF 2.0 Query/Views/Transformations (QVT)
- QVT aims to:
  - Query on MOF 2.0 models,
  - View on MOF 2.0 metamodels,
  - Transformation of 2.0 MOF models.
- QVT Final Adopted Specification
  - version: ptc/05-11-01
  - 23 pending issues: *http://www.omg.org/issues/*

Relationships between QVT metamodels



Java™ Architecture Analogy

- Relations language
  - Java language
- Core language
  - Java Byte Code
- Core semantics
  - the behavior specification for the Java Virtual Machine
- the standard transformation (from Relations to Core)
  - the specification of a Java Compiler which produces Byte Code

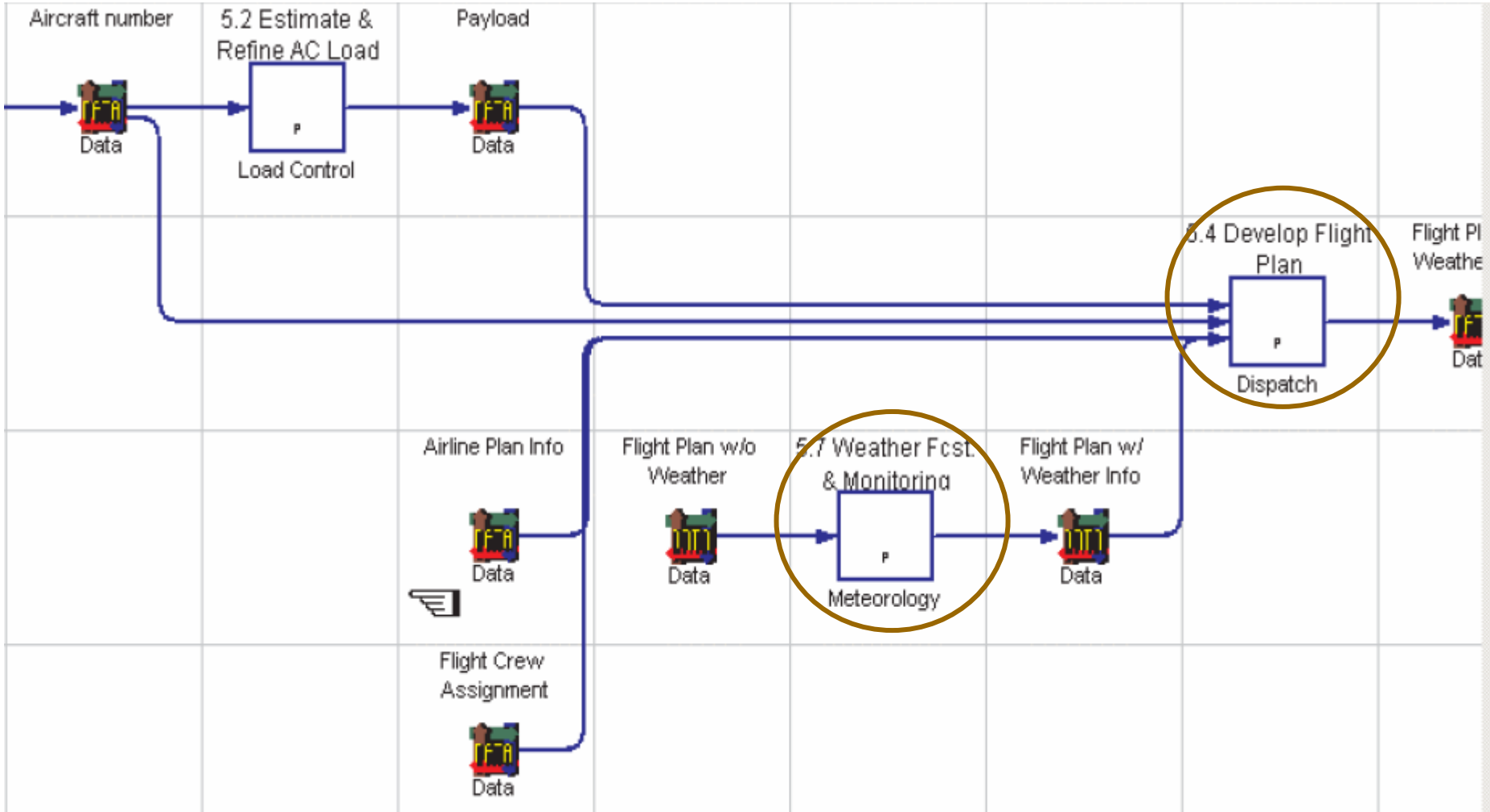
MDA practice

- Code generation based on templates
  - IBM Rational XDE
  - Codagen's Codagen Architect
  - ioSoftware's ArcStyler
- Modeling language extensions
  - IBM Rational Rose Technical Developer
  - Kennedy Carter's iUML
- Architected RAD solutions
  - IBM Rational Rapid Developer
  - Versata's Versata Logic Studio
  - M7's Application Assembly Suite
  - Kabira's Kabira Design Center
  - Compuware's OptimalJ

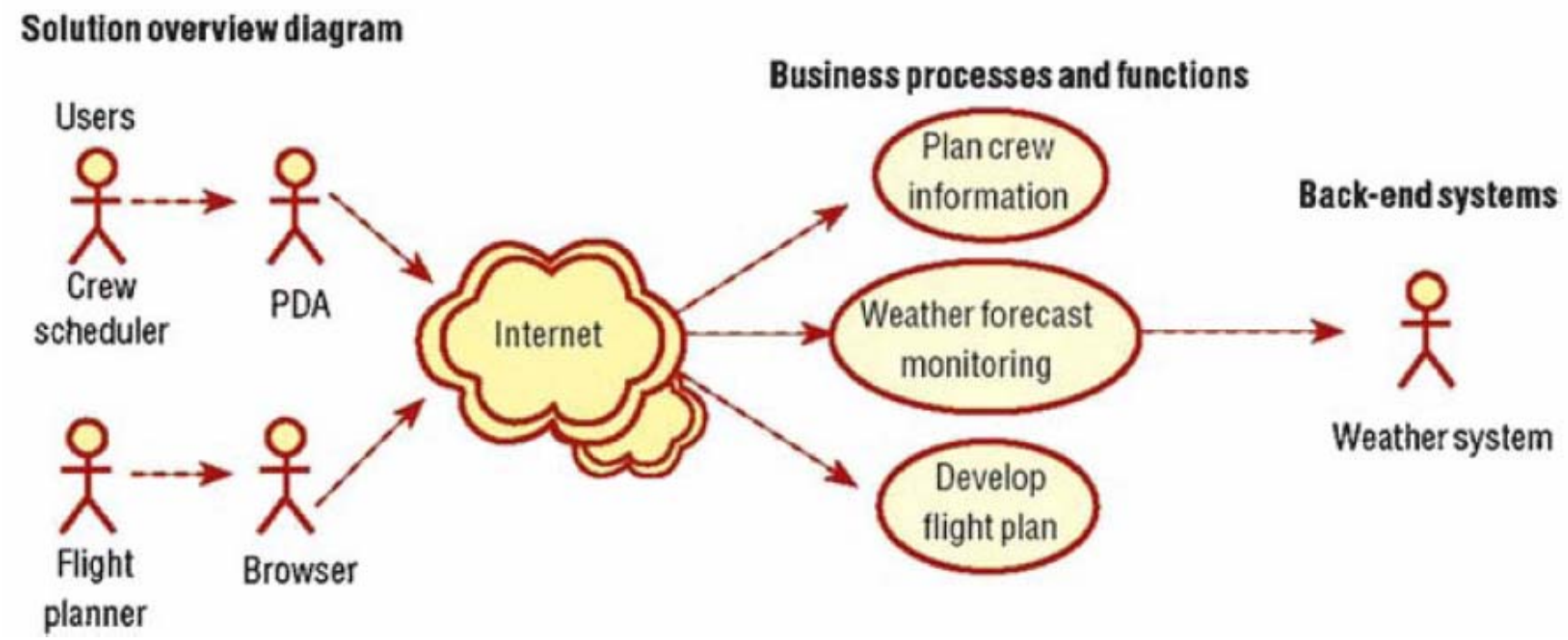
Examples

- Connecting business and IT
  - An important result of this real world example is that the client organization is able to gain visibility in the process of connecting domain-specific business needs into a technology-specific solution following a repeatable, predictable process.
- Platform abstraction

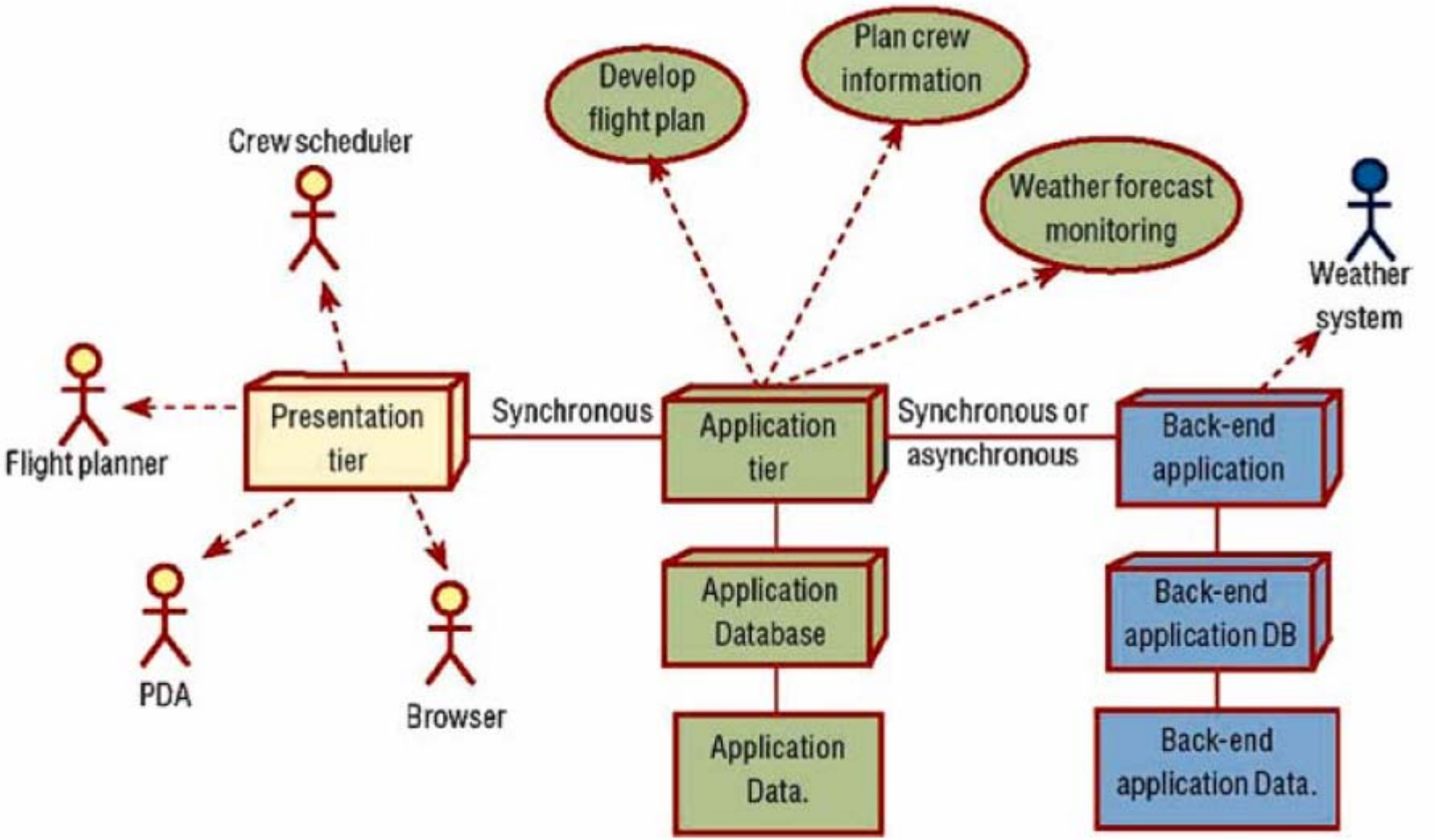
CIM modeling – IBM Rational Rose XDE



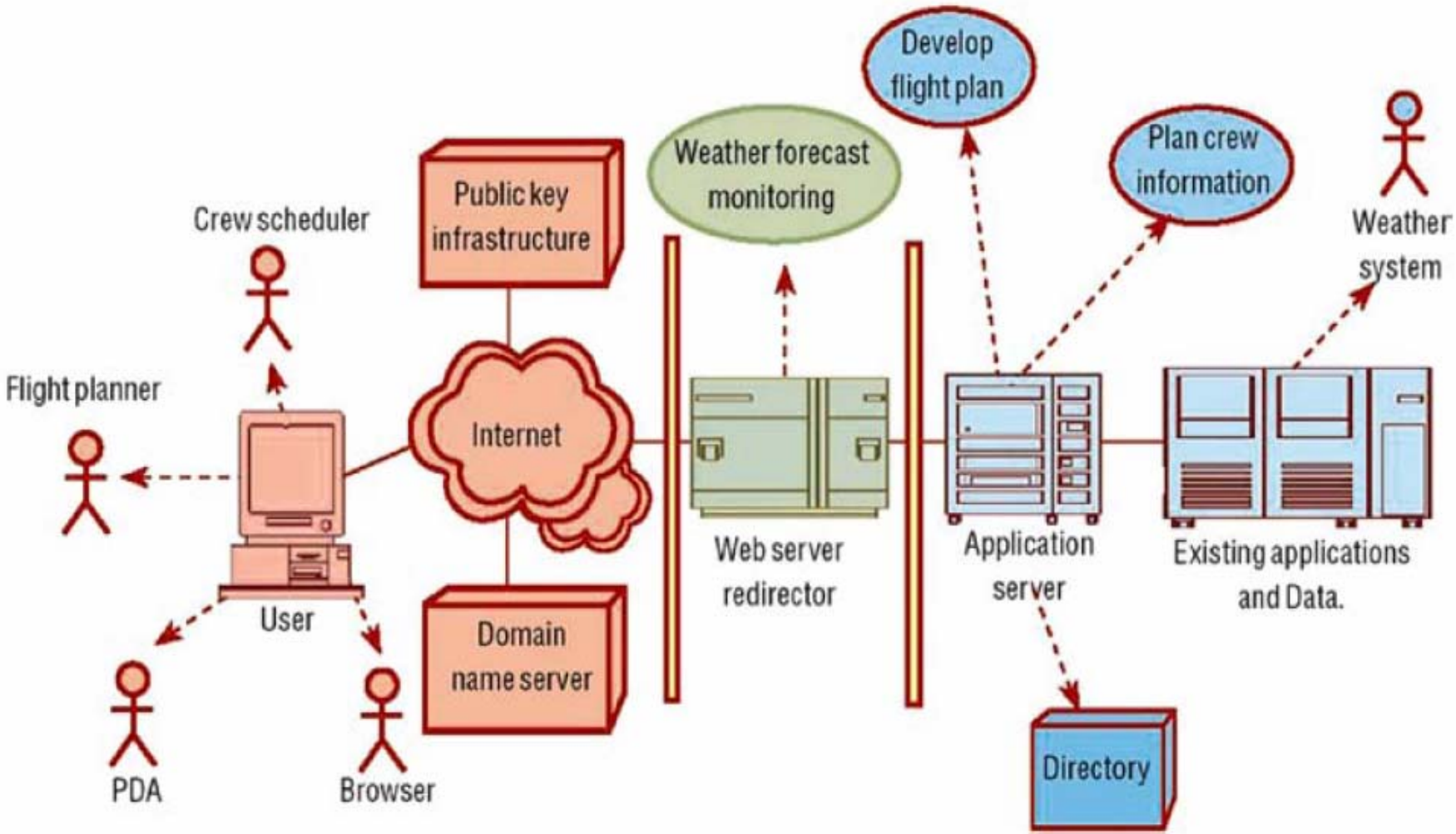
CIM → PIM


Solution overview diagram
Business processes and functions
Users
Crew scheduler
PDA
Internet
Plan crew information
Weather forecast monitoring
Develop flight plan
Back-end systems
Weather system
Flight planner
Browser

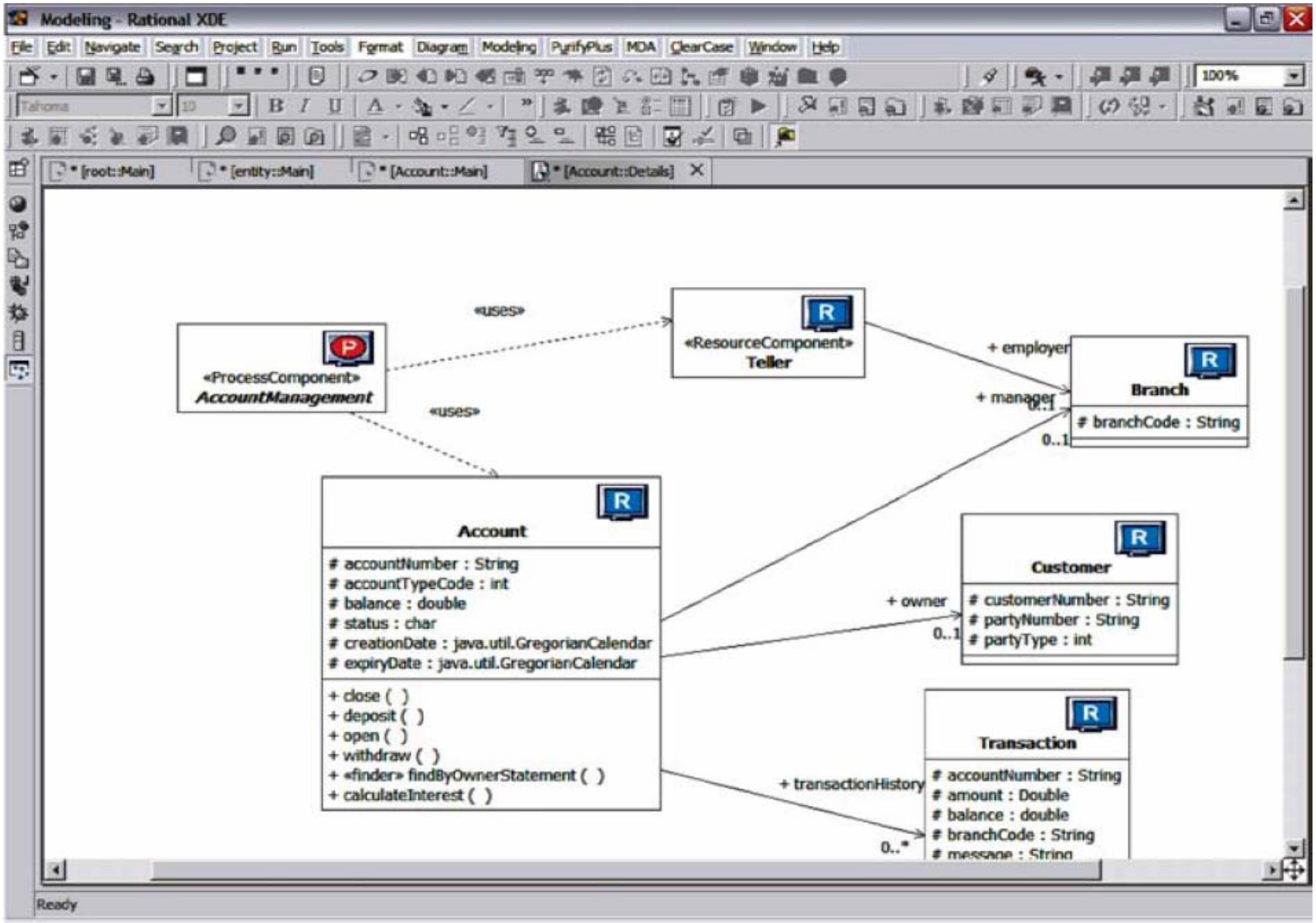
Refinement – using predefined patterns

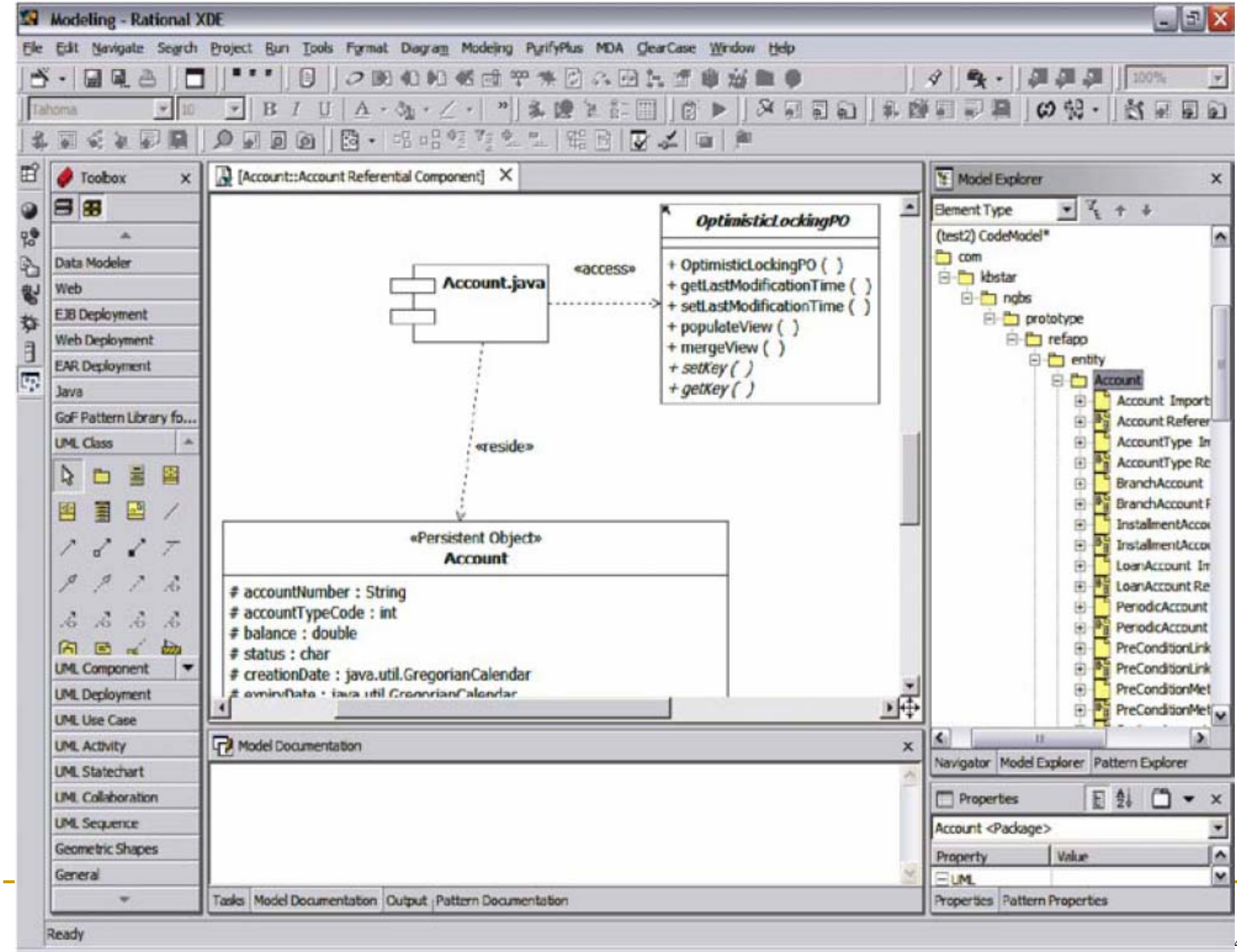


Refinement – using predefined patterns



Example 2: Platform abstraction




Modeling - Rational XDE
File Edit Navigate Search Project Run Tools Format Diagram Modeling PurifyPlus MDA ClearCase Window Help
Toolbox
Data Modeler
Web
EJB Deployment
Web Deployment
EAR Deployment
Java
GoF Pattern Library fo...
UML Class
UML Component
UML Deployment
UML Use Case
UML Activity
UML Statechart
UML Collaboration
UML Sequence
Geometric Shapes
General
[Account::Account Referential Component]
OptimisticLockingPO
+ OptimisticLockingPO ( )
+ getLastModificationTime ( )
+ setLastModificationTime ( )
+ populateView ( )
+ mergeView ( )
+ setKey ( )
+ getKey ( )
Account.java
«access»
«reside»
«Persistent Object»
Account
accountNumber : String
accountTypeCode : int
balance : double
status : char
creationDate : java.util.GregorianCalendar
Model Documentation
Tasks Model Documentation Output Pattern Documentation
Model Explorer
Element Type
(test2) CodeModel*
com
kbstar
ngbs
prototype
refapp
entity
Account
Navigator Model Explorer Pattern Explorer
Properties
Account <Package>
Property Value
UML
Properties Pattern Properties
Ready

Reference

- Bezivin J, Hammoudi S, Lopes D and Jouault F (2004). Applying MDA approach for Web service platform. Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International.
- AlanW Brown (2004). "Model driven architecture: Principles and practice." Softw Syst Model (2004).

Thanks