A presentation of MDD basics

Model-driven development (MDD) tutorial for managers

EUROPEAN SOFTWARE INSTITUTE,
Corporación Tecnológica Tecnalia
Parque Tecnológico, # 204
E-48170 Zamudio
Bizkaia (Spain)
www.esi.es

Context of this work

- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

Outline

- Presentation
- UML fundamentals
- MDA introduction
- Closing

Presentation

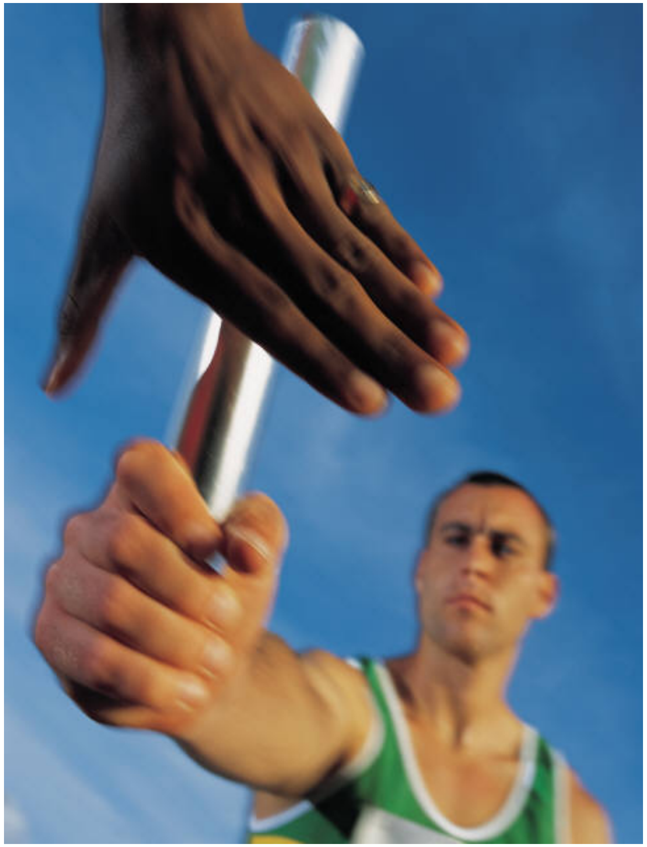
European Software Institute

- Non profit foundation
- Founded in 1993
- With European Commission, Basque Government and its partners and sponsors support
- Site: Zamudio, Bilbao, Spain
- www.esi.es

Presentation

Tutorial objectives

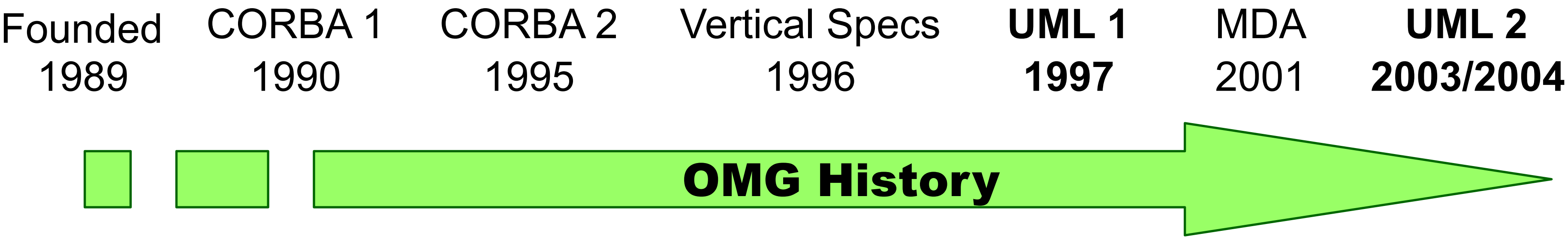
- Learn UML basic concepts
- Learn MDA basic concepts



UML fundamentals

UML and the OMG

- Unified Modelling Language is a standard of the **OMG (Object Management Group)** – http://www.omg.org
- UML current version: **version 1.5 – version 2.0**
- UML is used for representing **Software Systems Models**
- UML allows us to model different **software abstractions levels**: requirements, analysis, architecture, detailed design, ...



Founded 1989

CORBA 1 1990

CORBA 2 1995

Vertical Specs 1996

UML 1 1997

MDA 2001

UML 2 2003/2004

OMG History

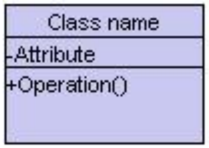
UML features

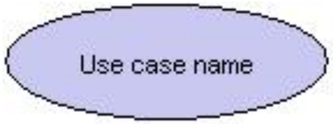
- Standard
- Many UML tools available
- Visual (and textual if desired)
- Used for modelling software
- Used for understand, design, maintain and control software application information
- Useful for other aims (for modelling business processes)

UML models

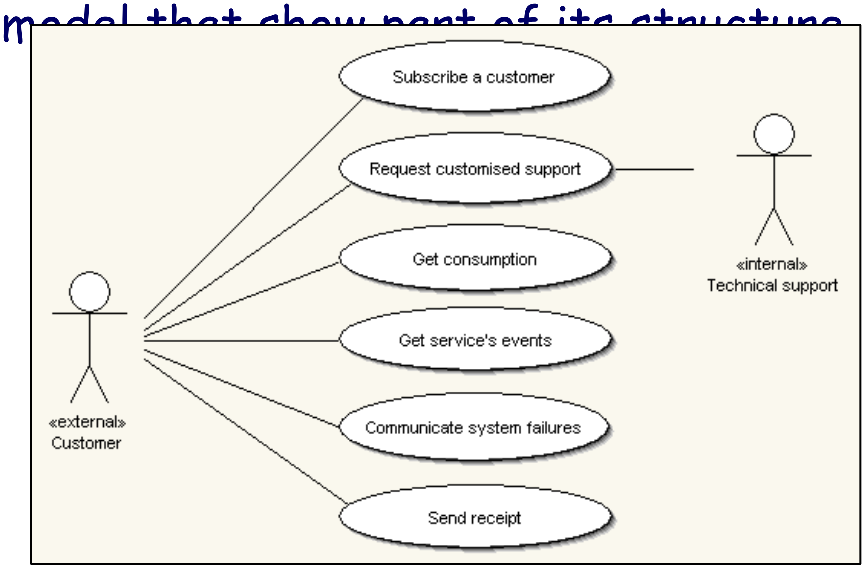
A UML model contains:

- **Elements:** classes, use cases, actors, interfaces, relationships, ...

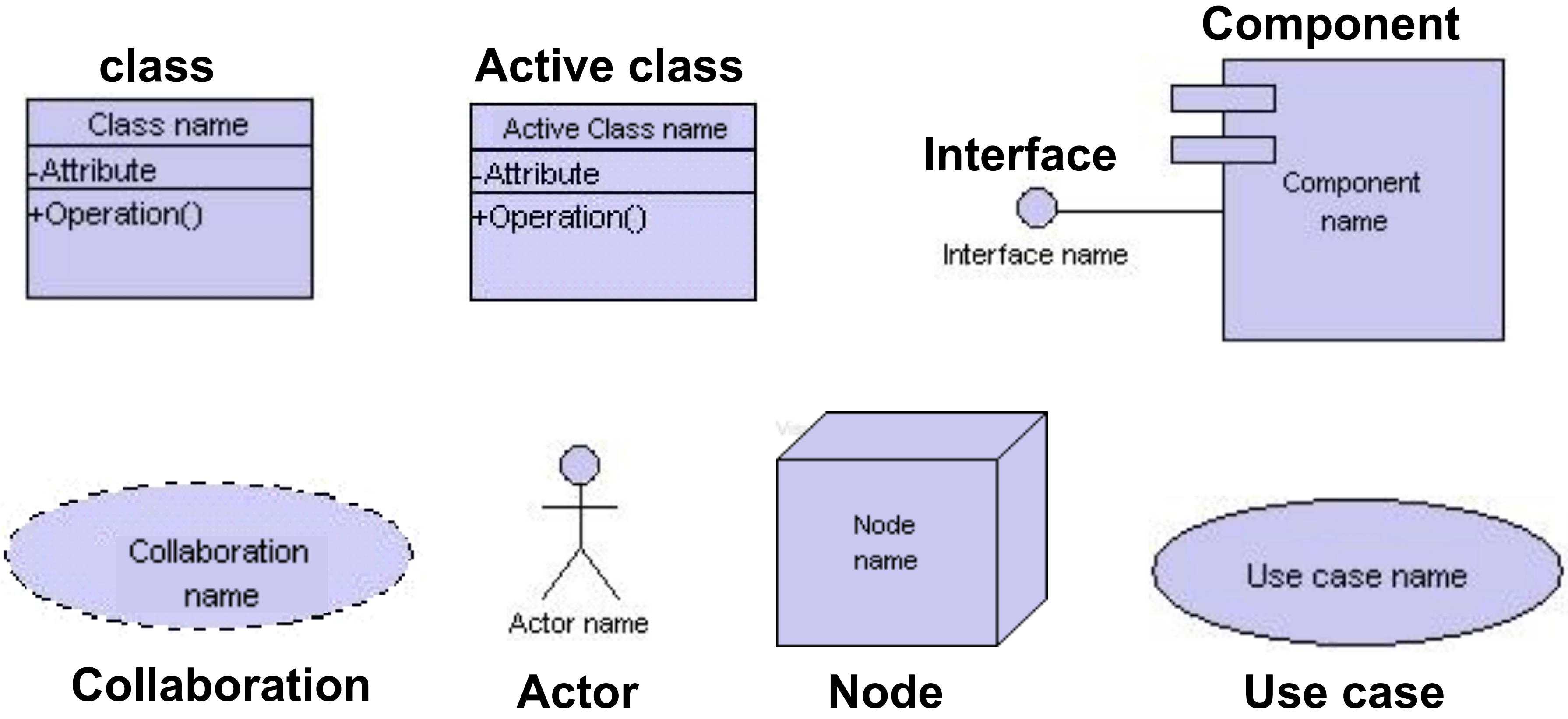




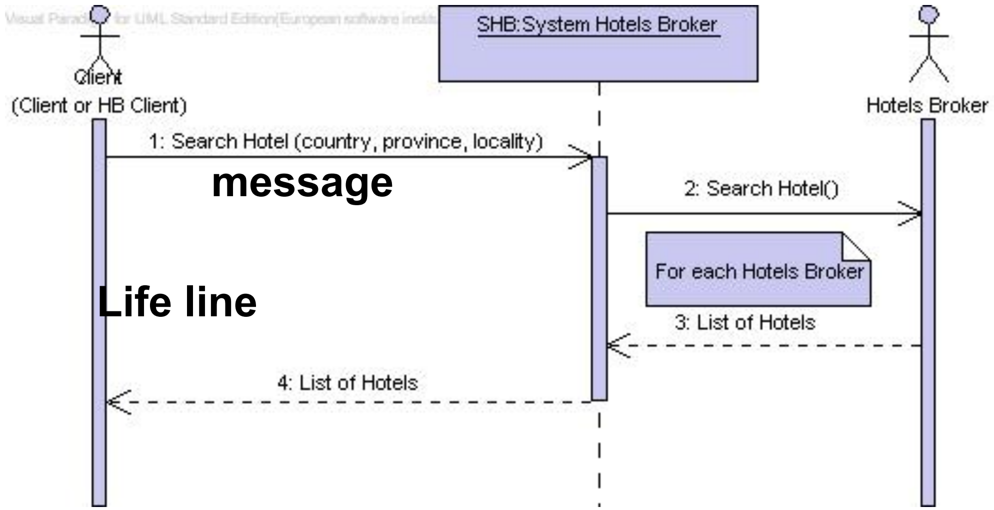
- **Diagrams:** views of the model that show part of its structure, behaviour and organization



Modelling in UML: Structural Elements



Modelling in UML: Behavioural elements

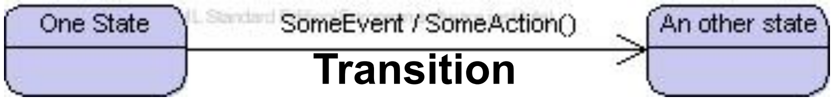


message

Life line

Interaction Diagrams

Event/Action

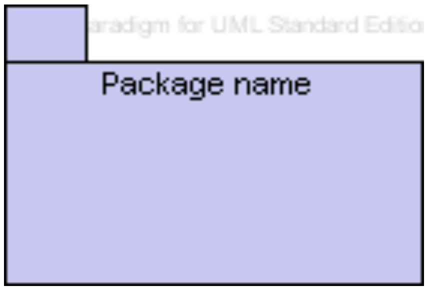


Transition

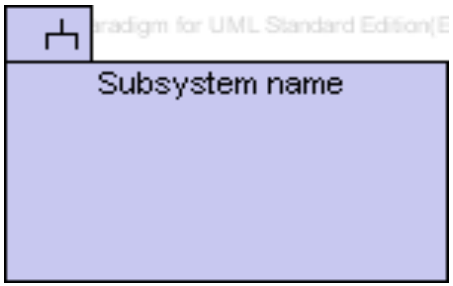
state

State machine
Activity Diagrams

Modelling in UML: Grouping elements



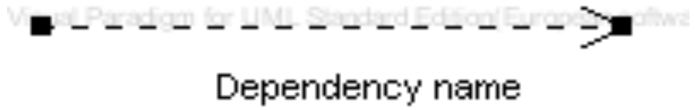
Package



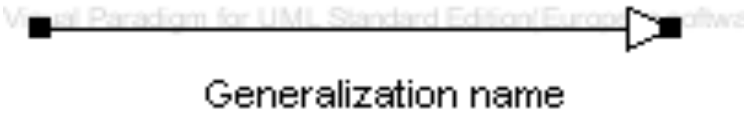
Subsystem

Modelling elements in UML: Relationships

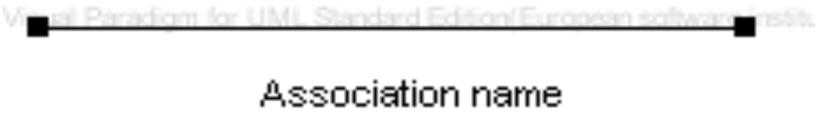
Dependency



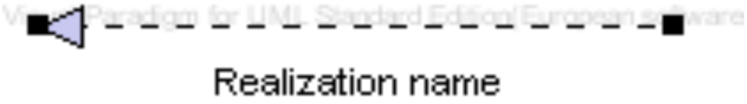
Generalization



Association

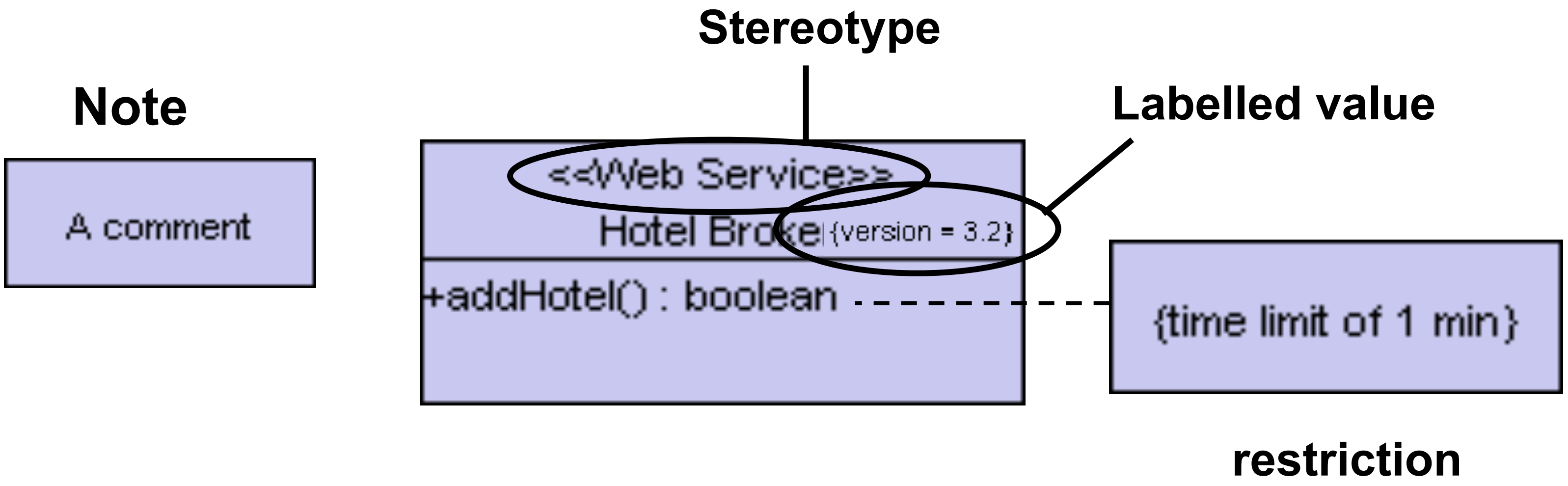


Realization



Other UML elements

- **Description Mechanisms:** Note
- **Extension Mechanisms:** Restriction, stereotypes and tagged values.

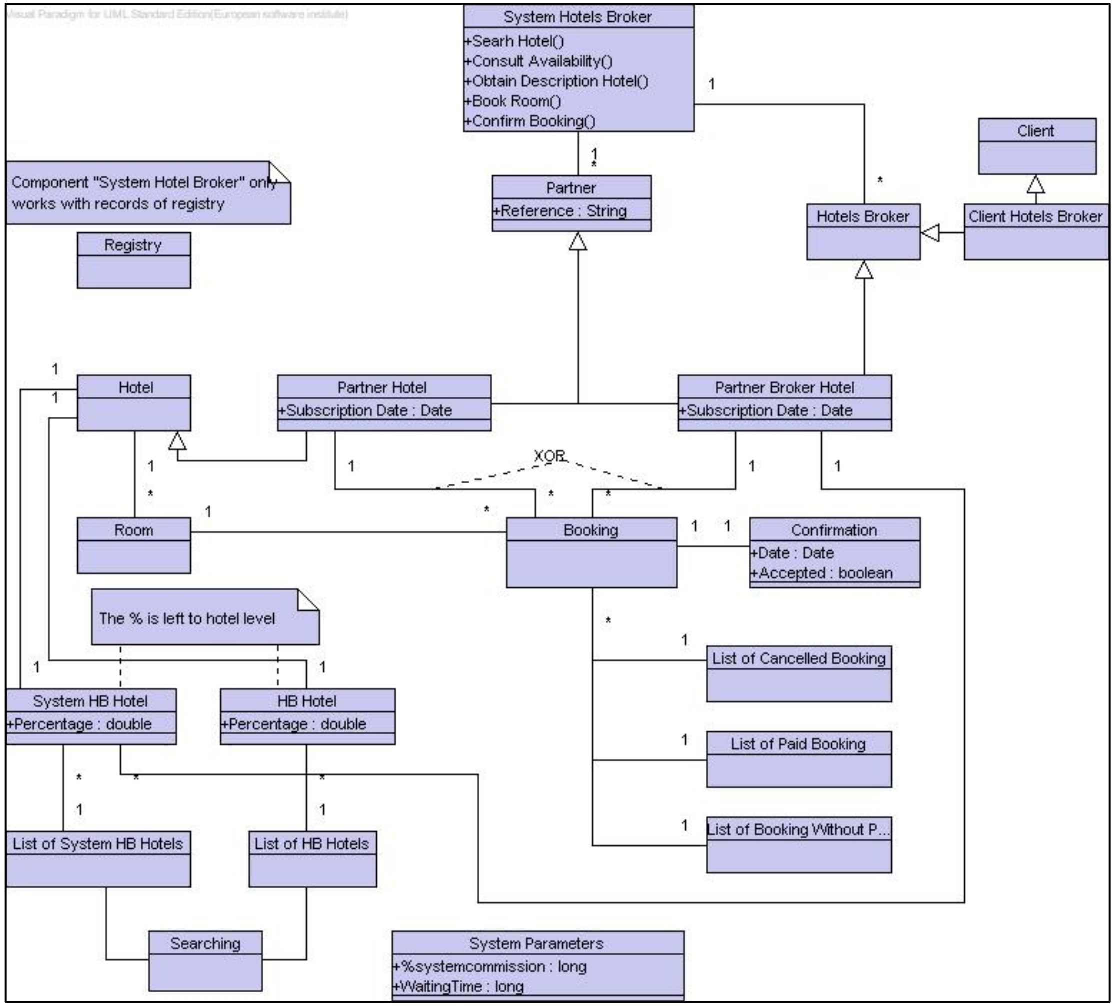


UML diagrams

- A diagram is a partial representation of the Model and must be consistent with the other views
- UML 1.5 defines 9 standard graphical diagrams:
  - use case diagram
  - class diagram
  - behavior diagrams:
    - statechart diagram
    - activity diagram
    - interaction diagrams:
      - sequence diagram
      - collaboration diagram
  - implementation diagrams:
    - component diagram
    - deployment diagram
  - Model management diagrams:
    - Class diagrams (using packages, sub-systems and models)

Examples of UML Diagrams

Class diagram

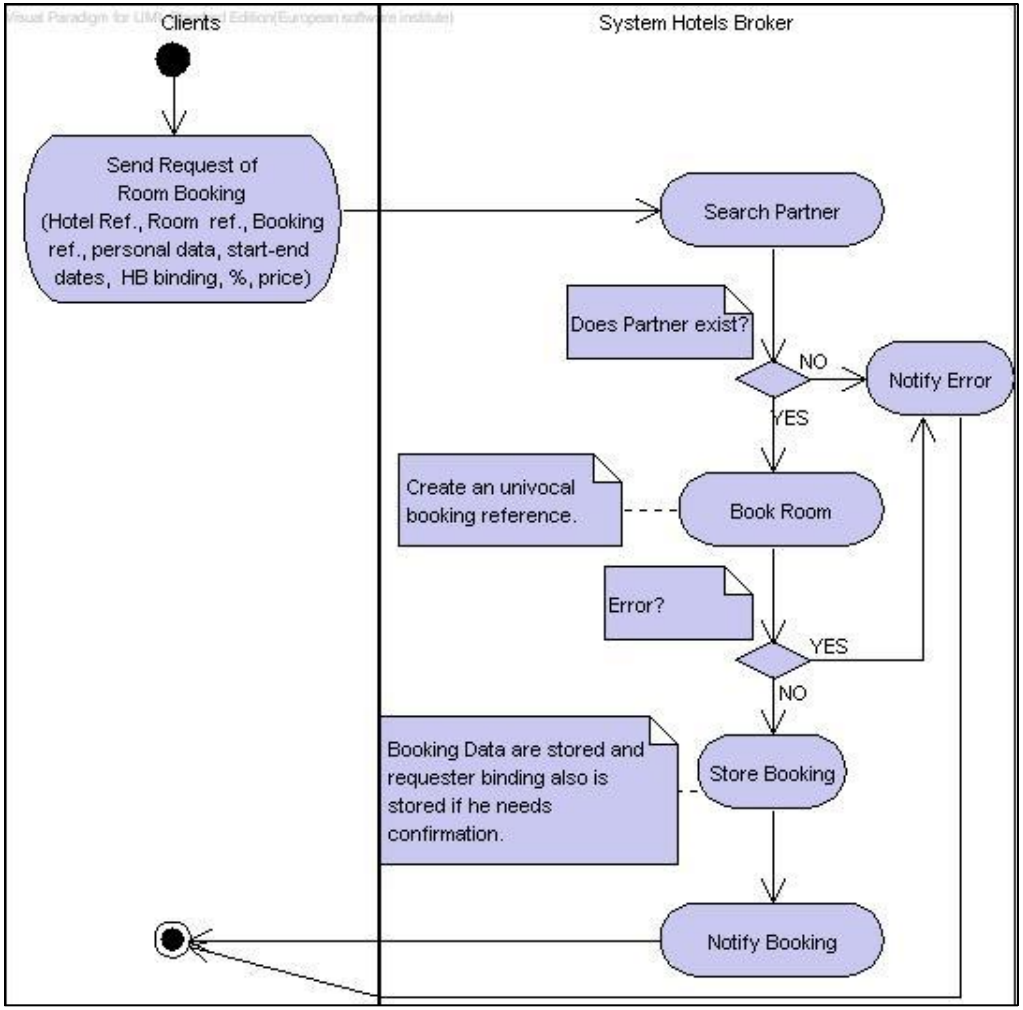


Component "System Hotel Broker" only works with records of registry

The % is left to hotel level

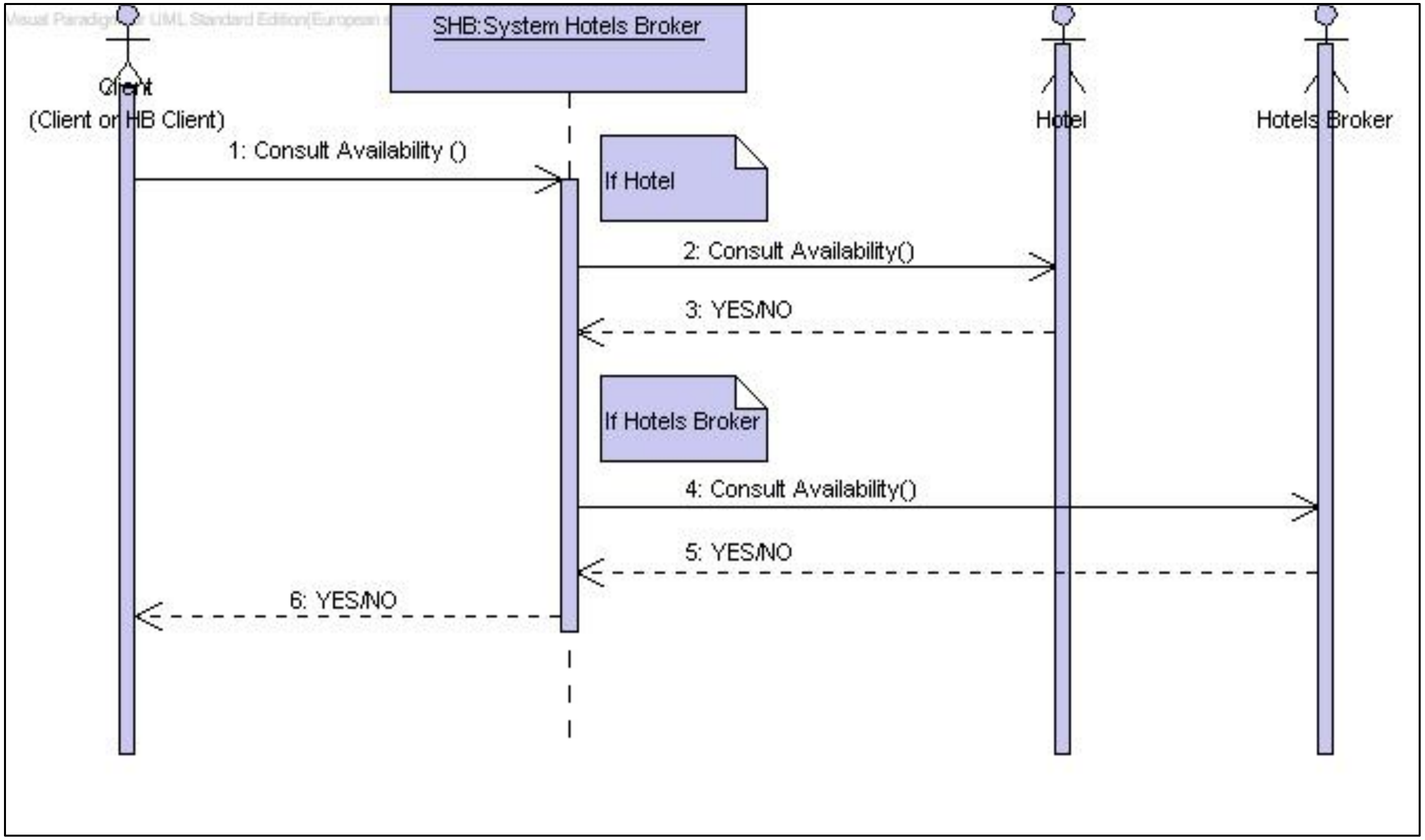
Examples of UML Diagrams

Activity diagram


Visual Paradigm for UML Standard Edition(European software institute)
Clients
System Hotels Broker
Send Request of Room Booking (Hotel Ref., Room ref., Booking ref., personal data, start-end dates, HB binding, %, price)
Search Partner
Does Partner exist?
NO
Notify Error
YES
Create an univocal booking reference.
Book Room
Error?
YES
NO
Booking Data are stored and requester binding also is stored if he needs confirmation.
Store Booking
Notify Booking

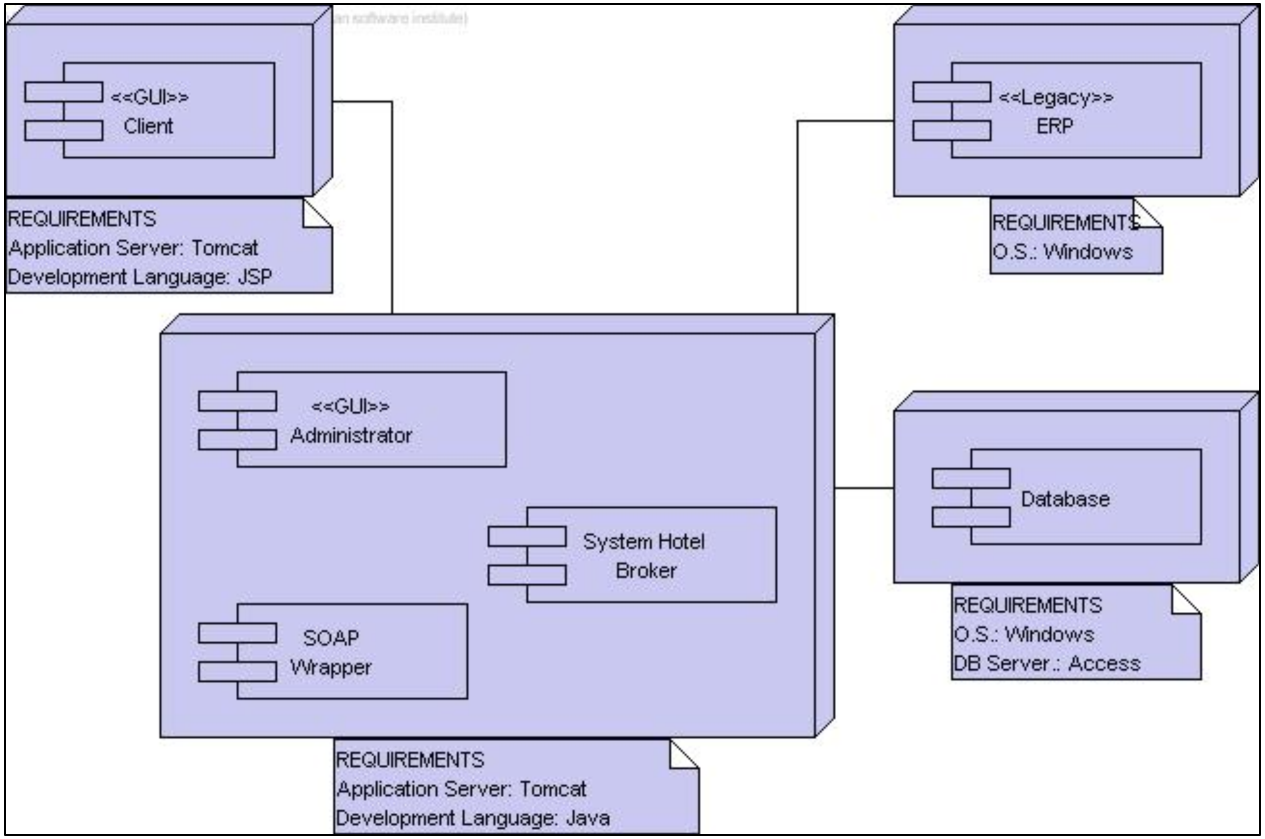
Examples of UML Diagrams

Sequence diagram


Visual Paradigm for UML Standard Edition(European
SHB:System Hotels Broker
Client
(Client or HB Client)
Hotel
Hotels Broker
1: Consult Availability ()
If Hotel
2: Consult Availability()
3: YES/NO
If Hotels Broker
4: Consult Availability()
5: YES/NO
6: YES/NO

Examples of UML Diagrams

Deployment diagram



REQUIREMENTS
Application Server: Tomcat
Development Language: JSP

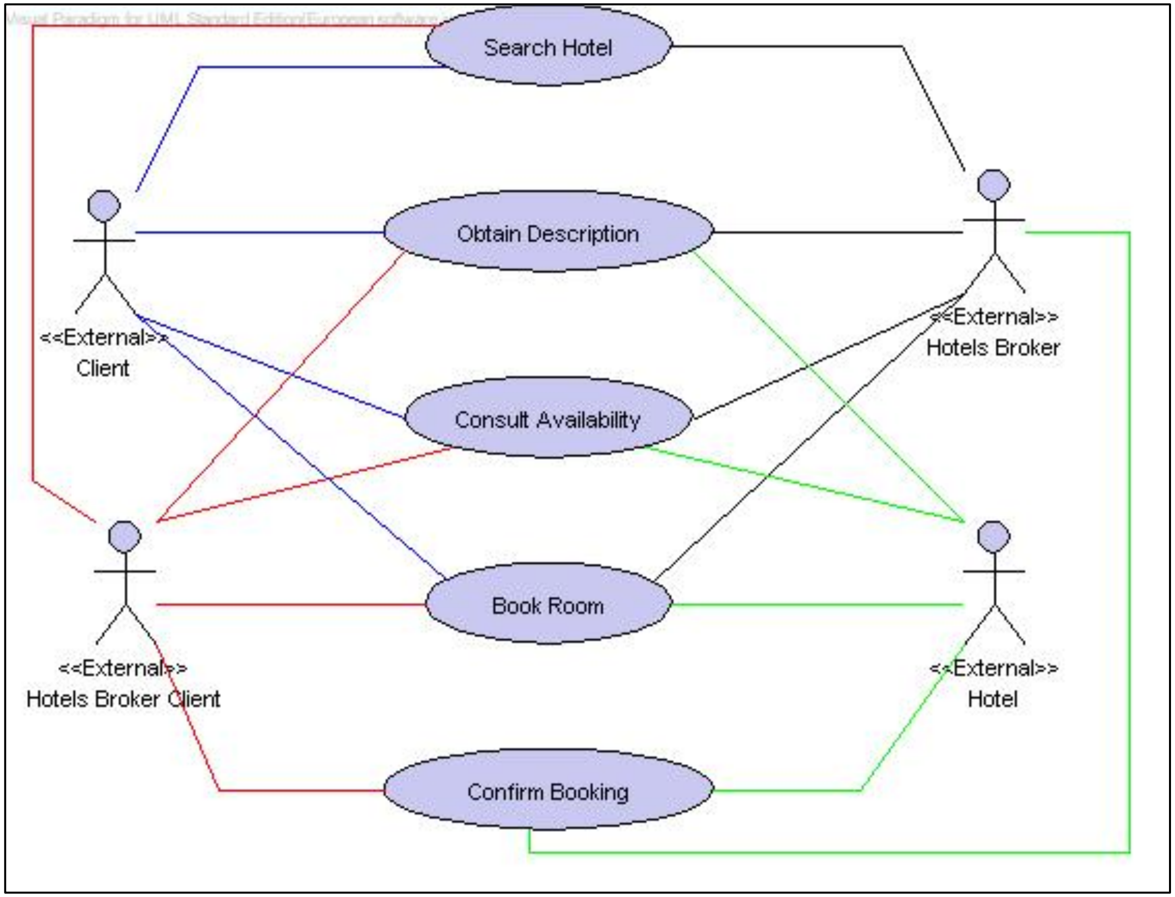
REQUIREMENTS
O.S.: Windows

REQUIREMENTS
O.S.: Windows
DB Server.: Access

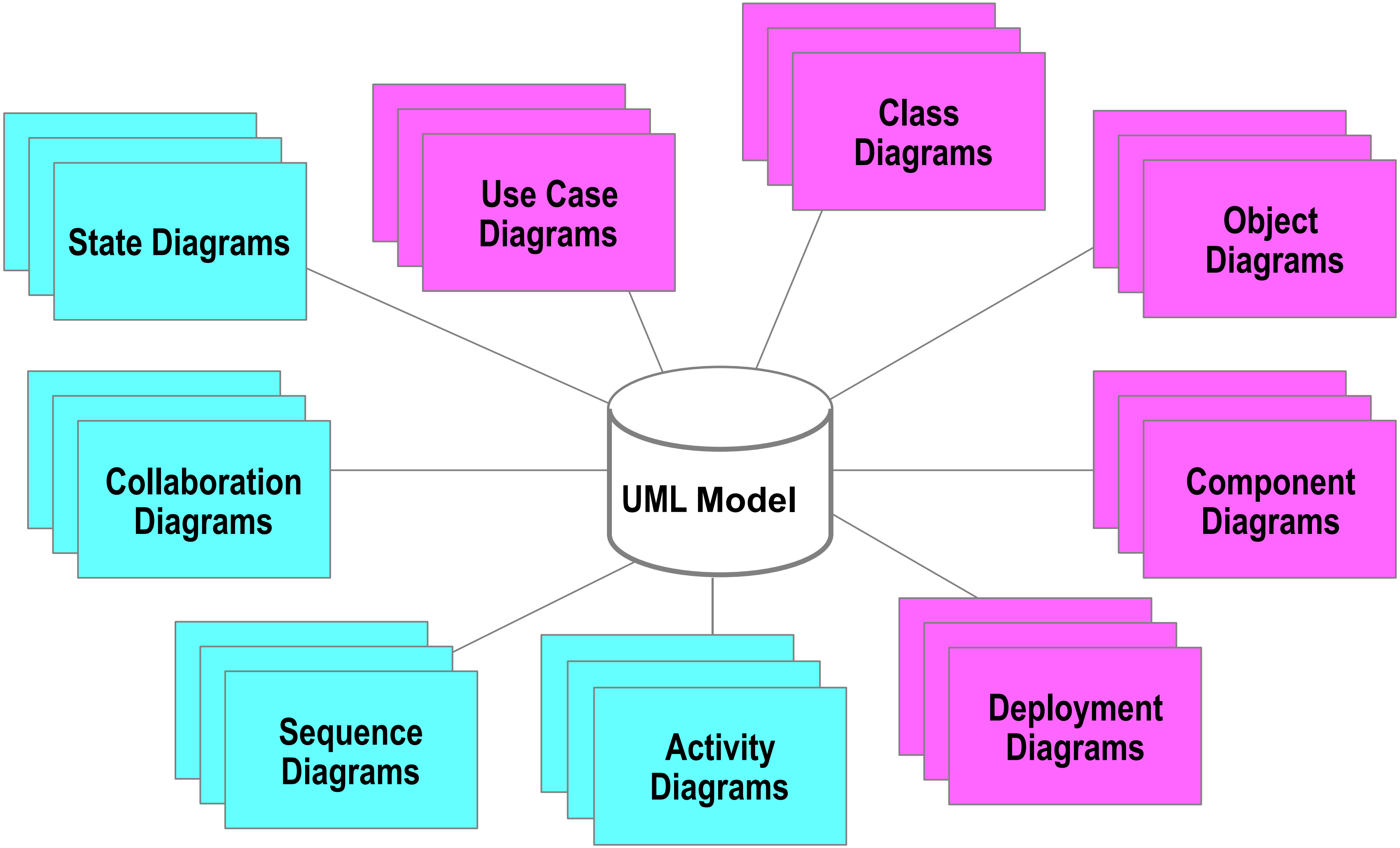
REQUIREMENTS
Application Server: Tomcat
Development Language: Java

Examples of UML Diagrams

Use case diagram



UML model


State Diagrams
Use Case Diagrams
Class Diagrams
Object Diagrams
Collaboration Diagrams
UML Model
Component Diagrams
Sequence Diagrams
Activity Diagrams
Deployment Diagrams

Exploring a UML model

There are two methods for exploring a UML model:

- **Browsing through its elements**
  - The elements are organized into packages (tree structure)
  - It is possible to navigate through the elements and analyse their relationships and characteristics
- ***Analyse its diagrams***
  - The diagrams provide views to understand the reality and the relationships between the elements of a model

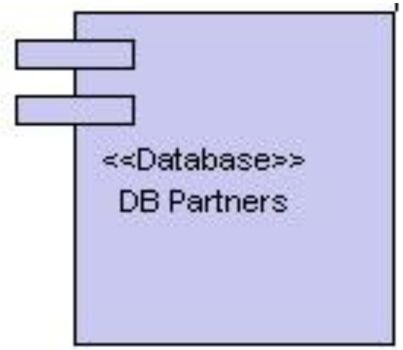
Extension mechanisms in UML

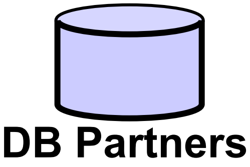
- They allow us to **adapt the UML language** to the needs of the **analysts** or the application **domain**

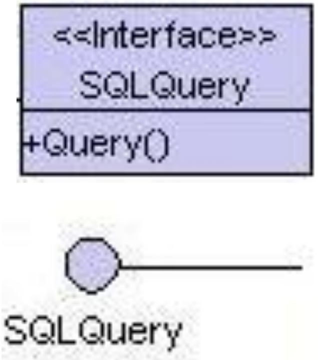
- There are three extension mechanisms:
  - Stereotypes
  - Restrictions
  - Labelled values

Stereotype

- **Extends** the vocabulary of **UML** with **new construction elements** derived from **existing** UML but specific to a problem domain
- Can have associated **restrictions and tagged values**
- Possibility of assigning an **icon** for a better graphical representation



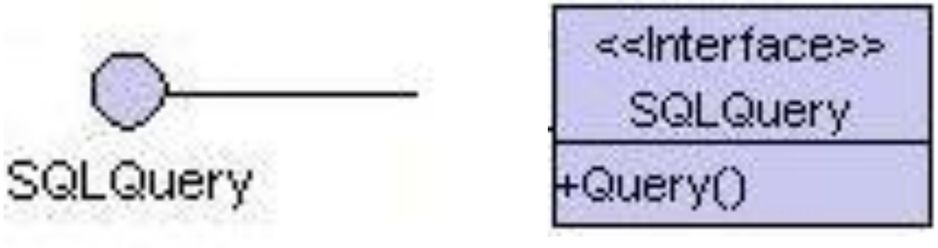




Restriction

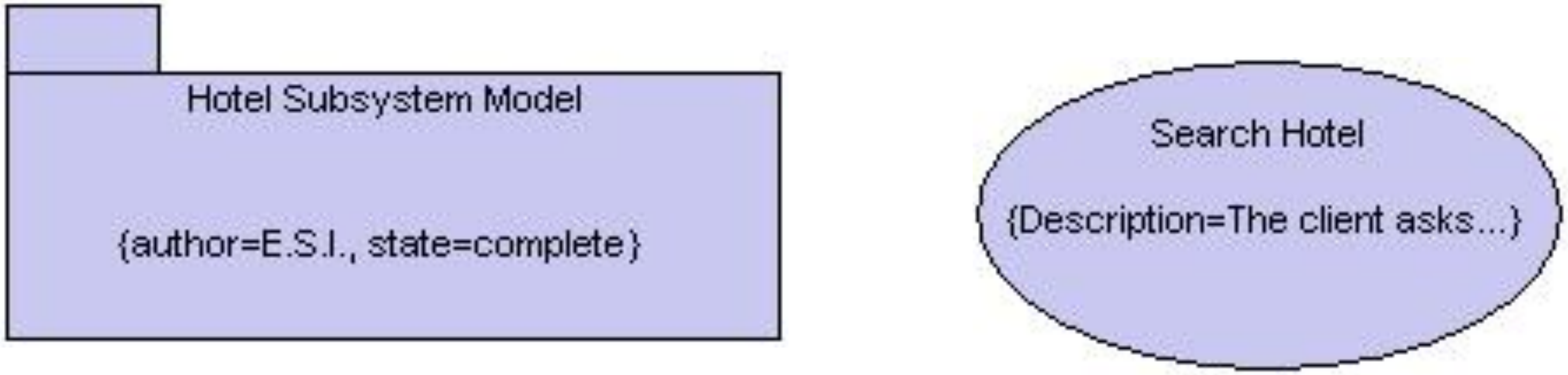
- Is a semantical **condition** represented by a **textual expression**
- Imposes some kind of condition or **requisite on the element** to which it is applied
- OCL – Object Constraint Language

{An interface does not have attributes, only operations}



Tagged value

- Is a **property** associated to a **model element**
- Used to store **information** about the element
  - Management information, documentation, coding parameters, ...
- Generally, the **tools** store this information but **it is not shown in the diagrams**


Hotel Subsystem Model
{author=E.S.I., state=complete}
Search Hotel
{Description=The client asks...}

UML profile: "Your language"

- A set of defined extensions which can be **reused** in various models
- A set of **stereotypes, tagged values and restrictions** which adapt UML with a specific goal in mind:
  - Adjusting UML for a specific **domain**, representing the domain's concepts through the use of the extension mechanisms
  - **Generate** code and documentation
  - Perform **Model transformations** (refinement)
- Tools exist which are capable of managing (creating and using) UML profiles

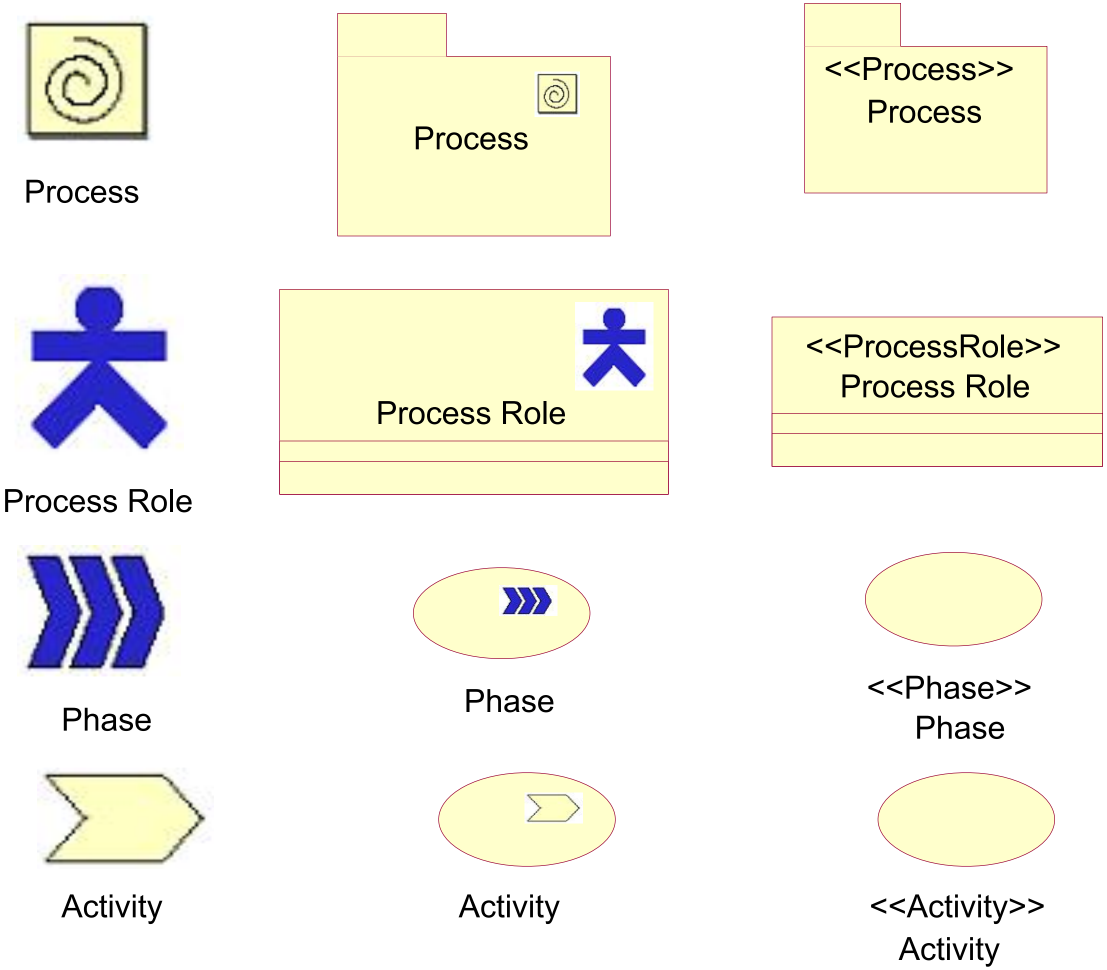
UML profile example: SPEM (1/3)

SPEM: Software Process Engineering Metamodel

Meta-model and UML profile to describe software engineering processes

- Identifies the typical concepts of a process (process, phase, role, model, etc.)
- Defines them using UML extensions (stereotypes applied to various elements: class, use cases, operations, etc.)
- Assigns a characteristic icon to each new item.

UML profile example: SPEM (2/3)



Process

Process

<<Process>>
Process

Process Role

Process Role

<<ProcessRole>>
Process Role

Phase

Phase

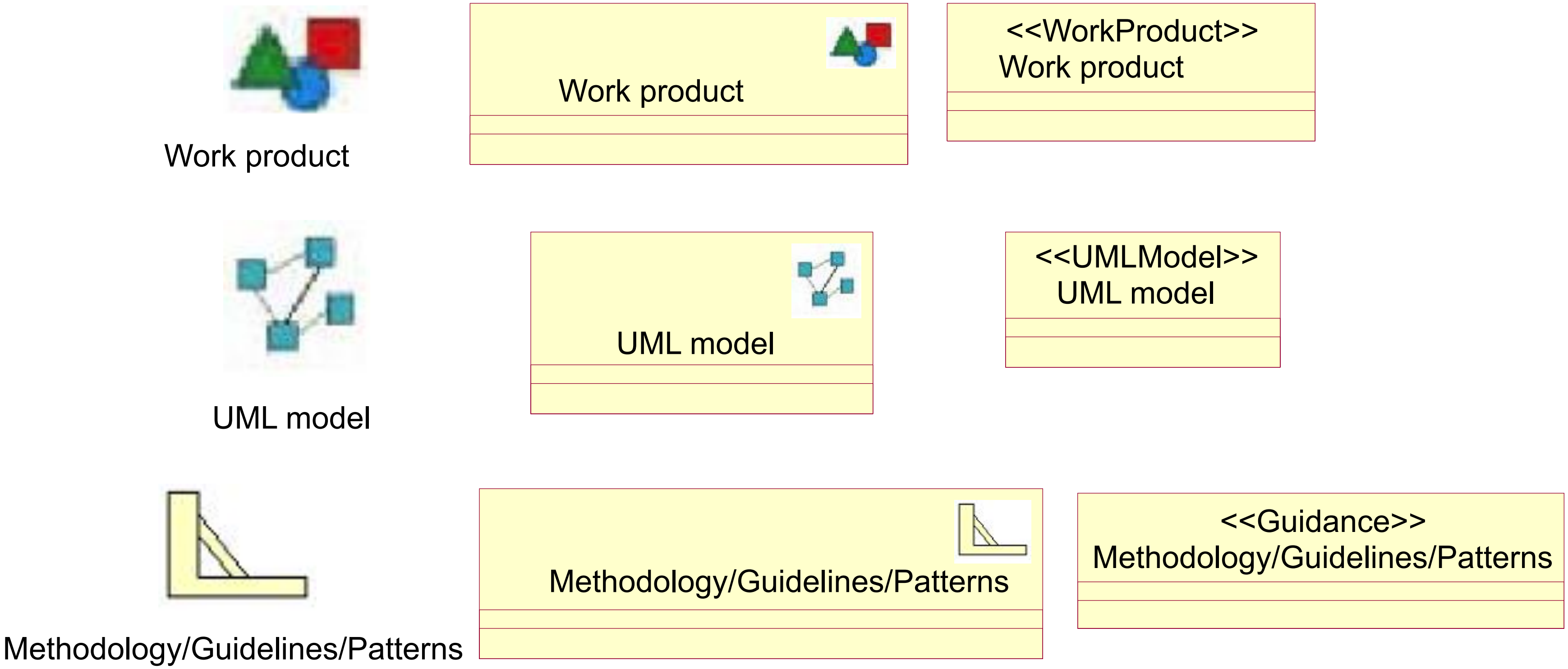
<<Phase>>
Phase

Activity

Activity

<<Activity>>
Activity

UML profile example: SPEM (3/3)



Work product

Work product

<<WorkProduct>>
Work product

UML model

UML model

<<UMLModel>>
UML model

Methodology/Guidelines/Patterns

Methodology/Guidelines/Patterns

<<Guidance>>
Methodology/Guidelines/Patterns

Why model?

- Models are used by software professionals to **communicate** their work and their knowledge to clients, developers, manager, etc.
  - System and functional requirements established by the client
  - Structure and design of the software solution
  - The relationship between a requirement and the code
  - Progress made
- UML models are appropriate for **documenting** software applications (requirements, analysis, architecture, detailed design, test cases)

Visual modelling benefits

- Improves **communication** reducing cost caused by incorrect interpretation
  - Internally in work groups
  - Externally with partners and clients
- Improves **maintenance**, eases **evolution**
- Allows better management of **complexity** through separation of concerns in different diagrams
- Increases **visibility** in software projects
- Strengthens **reuse** at design time

Evolution of visual modelling to a model-driven design

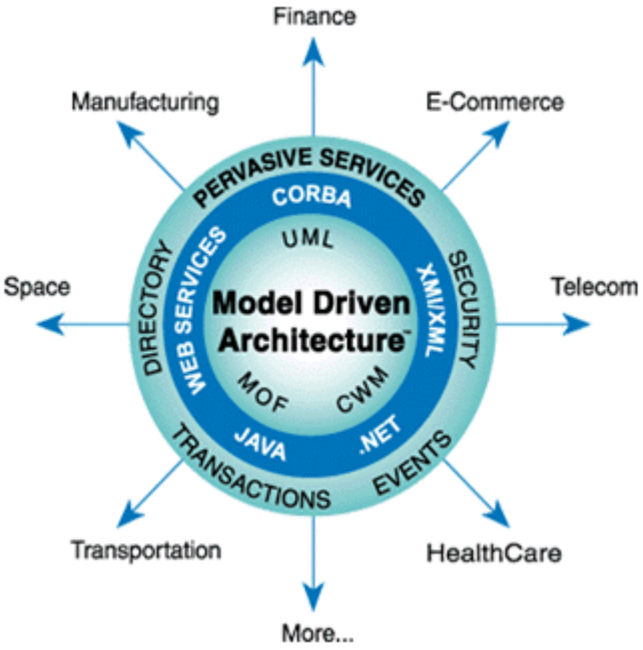
- Systems modelling has, until now, used the traditional methods of systems development as their starting point. Giving rise to the following situations:
  - 1 analysis -> n developers – n different systems
  - Development of designs starting from scratch or in the “best” case reusing existing designs on an ad-hoc basis.
  - Knowledge of business processes distributed amongst the various analysts
  - 1 Problem – 1 new systems development

Evolution of visual modelling to a model-driven design

- UML is not the solution to the problems we've just stated
- We need an approach in which the knowledge acquired by a company through its entire life to be in collected and stored in one place
- We need to have business logic available and accessible to ease the development of new solutions
- We need to provide mechanisms that allow organisations to adapt easily to technological changes and shifts

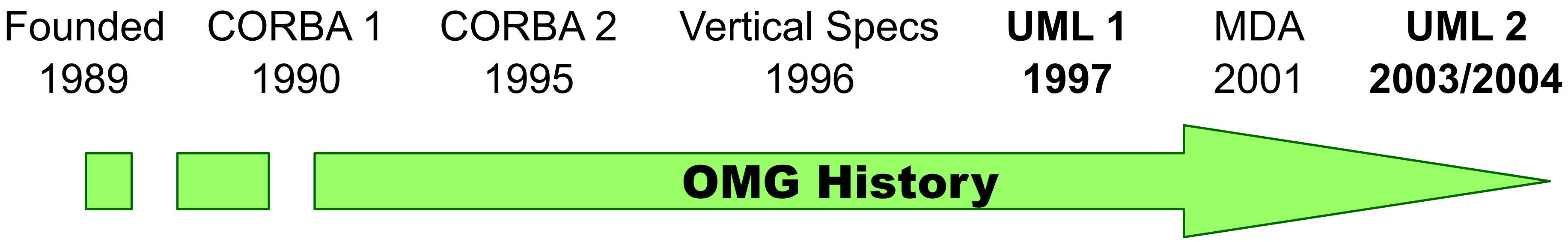
MDA

MDA introduction



MDA and the OMG

- Just like UML, MDA is a standard promoted by the OMG
- It is a new way to focus on software development and is based on models
- Adoption was started in 2001



Founded	CORBA 1	CORBA 2	Vertical Specs	**UML 1**	MDA	**UML 2**
1989	1990	1995	1996	**1997**	2001	**2003/2004**

OMG History

What is MDA and what does it seek?

- MDA is a new way to look at software development, from the point of view of the models.
- **Separates** the operational specification of a system from the details such as how the system uses the platform on which it is developed.
- MDA provides a means to:
  - **Specify** a system independently of its platform
  - Specify platforms
  - Chose a platform for the system
  - **Transform** the system specifications into a platform dependent system
- Three fundamental objectives: portability, interoperability and reuse.

MDA fundamentals

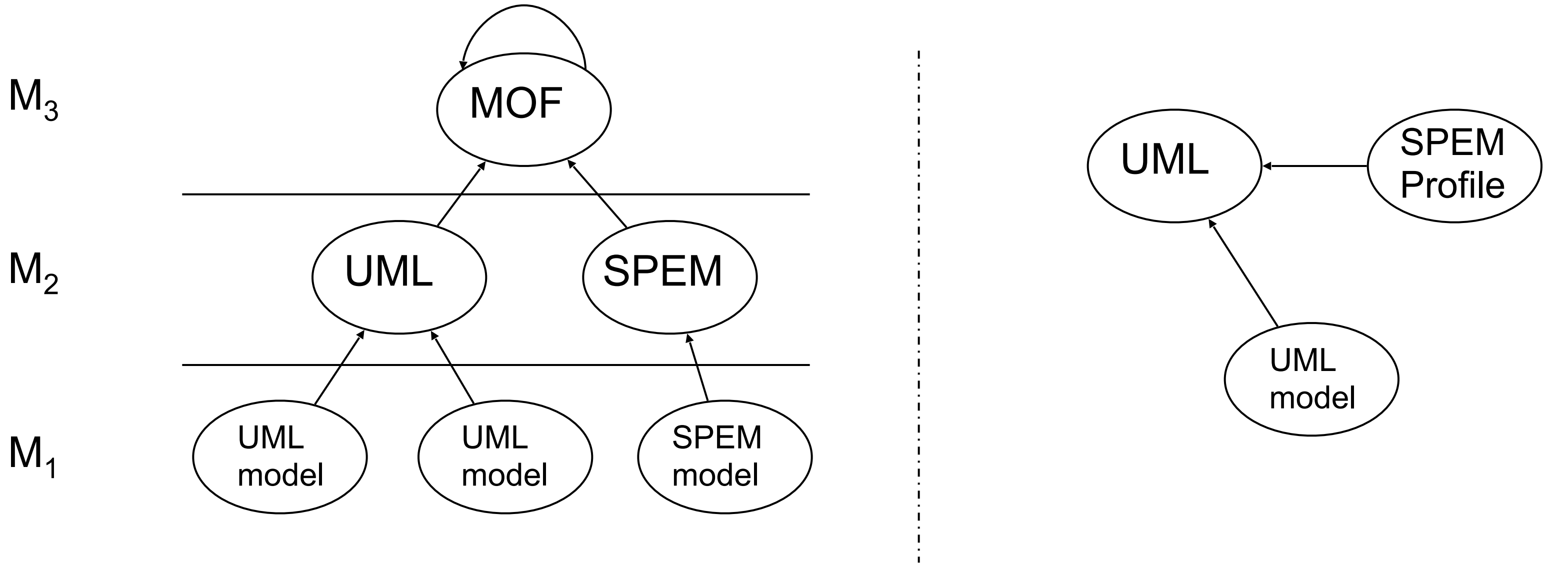
- Abstraction:
  - CIM: Computation Independent Model
  - PIM: Platform Independent Model
  - PSM: Platform Specific Model
- Transformations:
  - Between different levels of abstraction
  - Enriched models: notes, composition,…
- Everything is a Model:
  - Metamodel and Meta-metamodel = Models of Models

Benefits of MDA

- Allows **implementation flexibility** regarding platform choice. Reducing the impact of technological changes.
- **Reuse**.
- Improves software development **process**:
  - Expressing the solution in terms of the domain specific problem.
  - Earlier detection of problems.
  - **Automation** of parts of the development process.
- Improves development **maintenance**: Models are an active part of the design process not solely documentation.
- Eases requirement **traceability**:
  - Improving change control
  - Improving solution validation

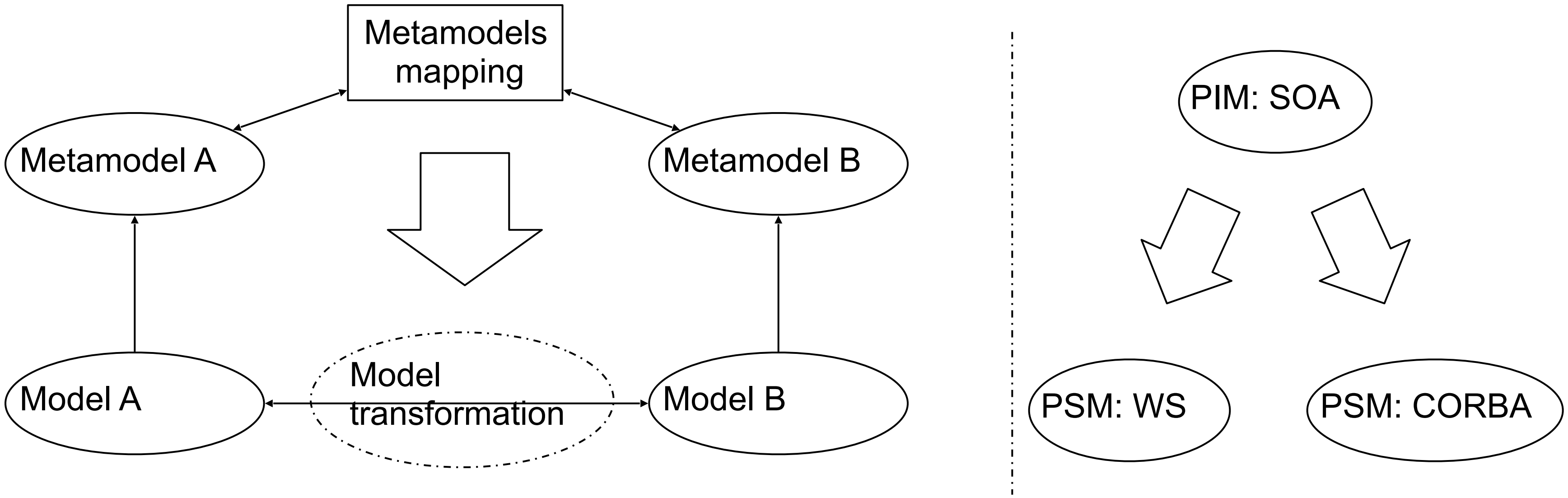
MDA basic elements (1/3)

- MODELS: cornerstone of MDA.
  - Metamodels: everything is a model. MOF. EMF (Eclipse).
  - UML profiles: Adapted modelling language.



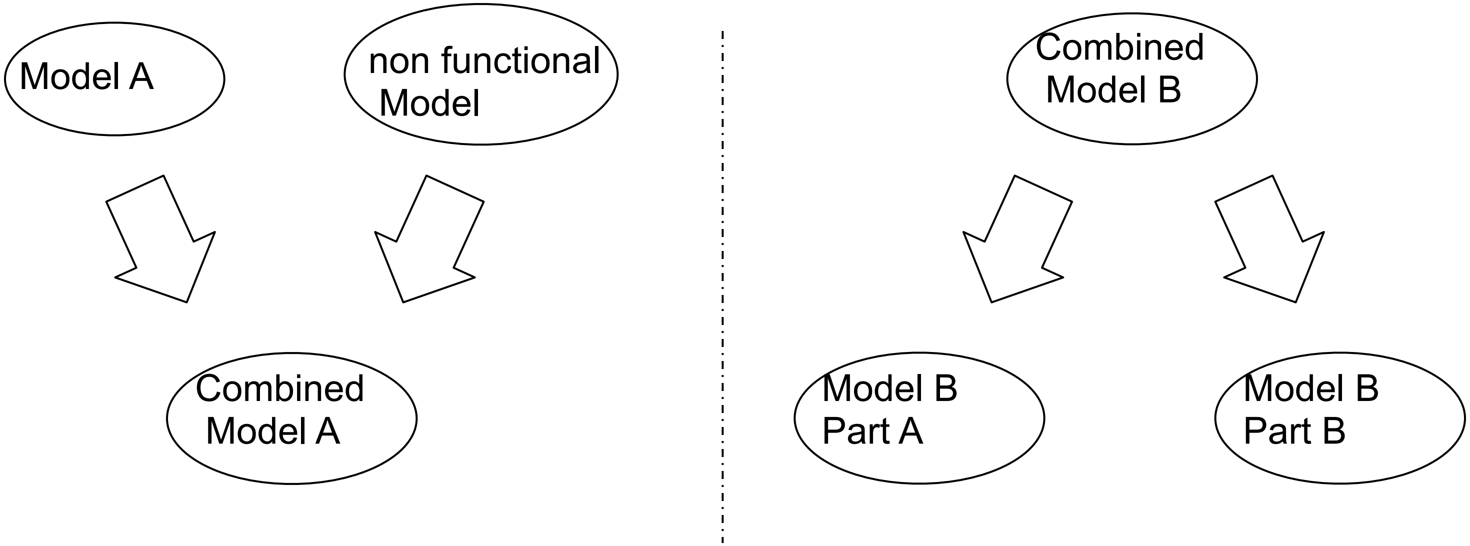
MDA basic elements (2/3)

- Transformations
  - Models with notes
  - Metamodels mapping
  - MOF QVT
  - Code generation: transformation

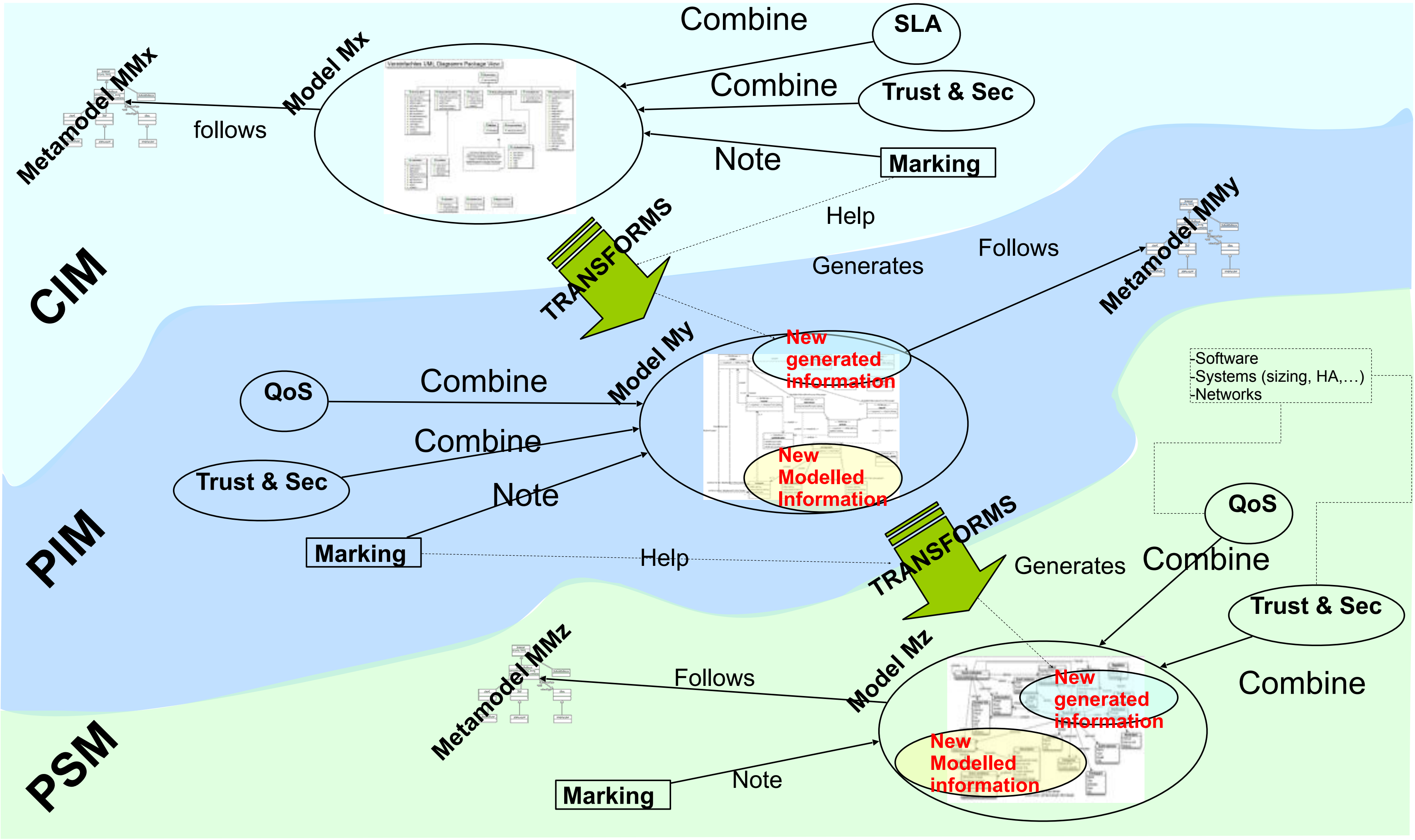


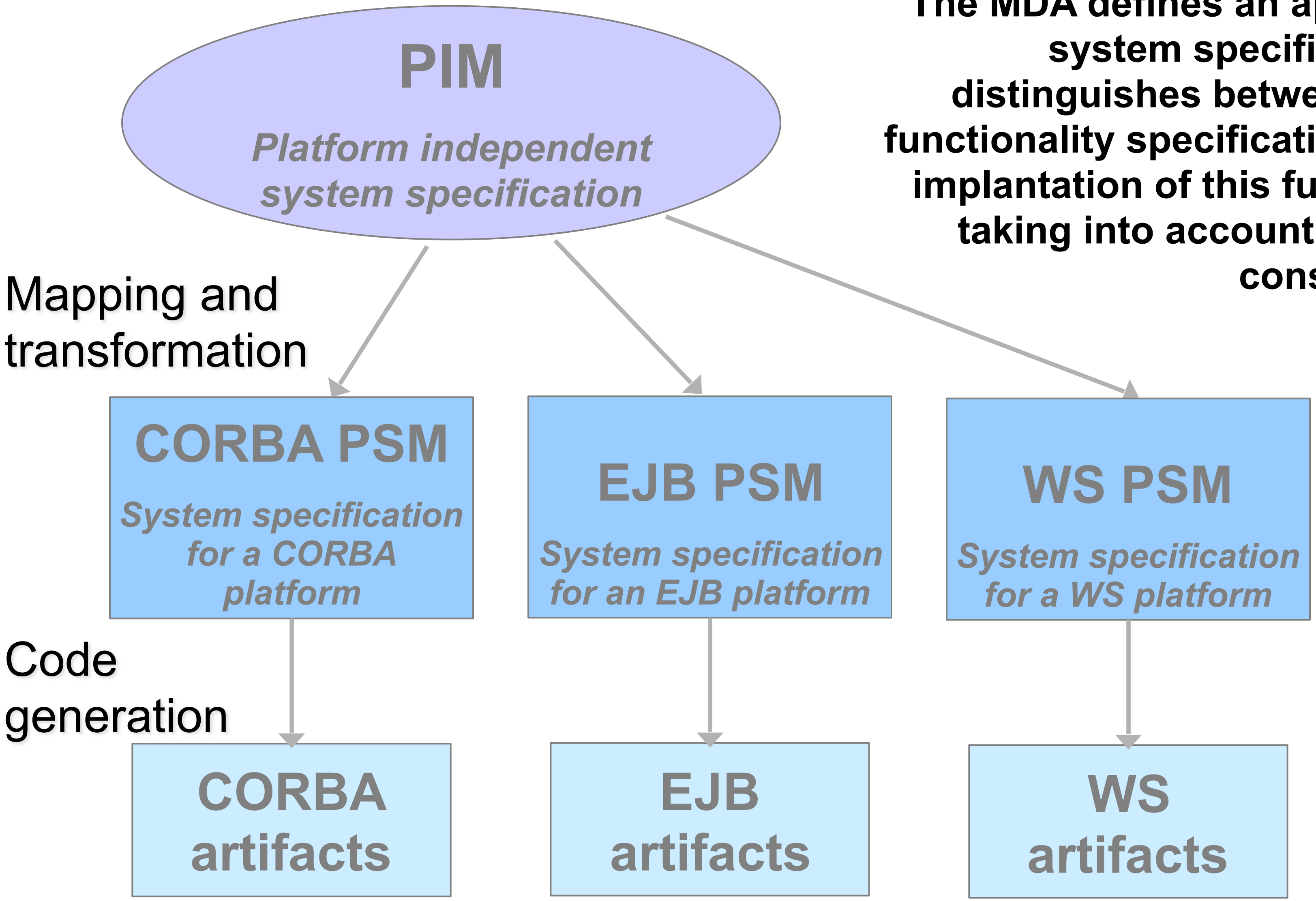
MDA basic elements (3/3)

- Model composition
  - Composite solutions (federated systems, multiplatform systems,...)
  - Non functional aspects



MDA Perspective





PIM

Platform independent system specification

Mapping and transformation

CORBA PSM

System specification for a CORBA platform

EJB PSM

System specification for an EJB platform

WS PSM

System specification for a WS platform

Code generation

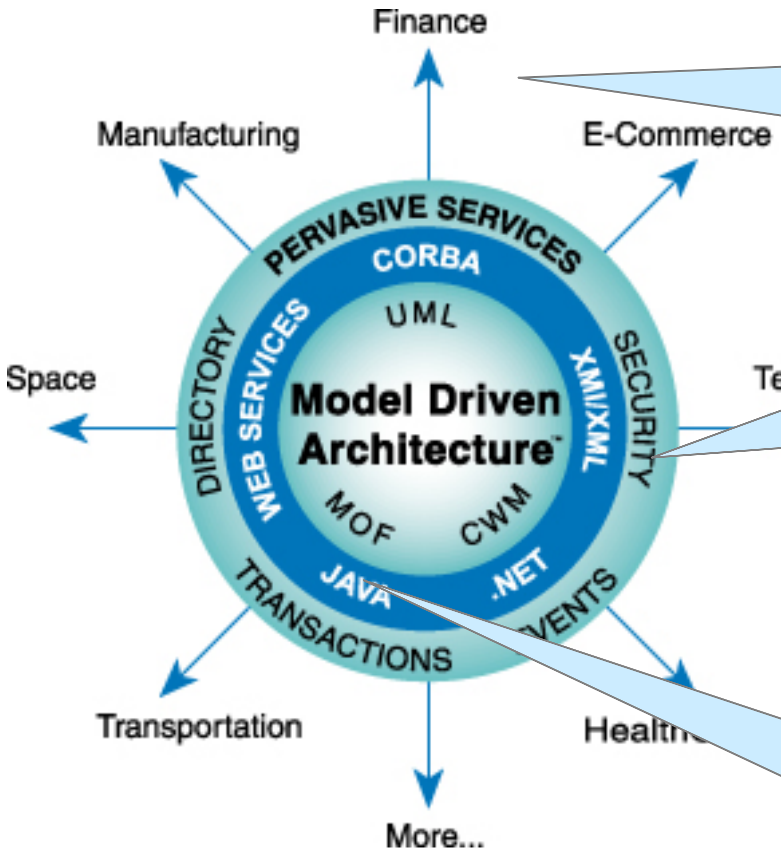
CORBA artifacts

EJB artifacts

WS artifacts

“The MDA defines an approach to system specification that distinguishes between system functionality specification and the implantation of this functionality taking into account a platform consideration”

How to advance in the MDA adoption



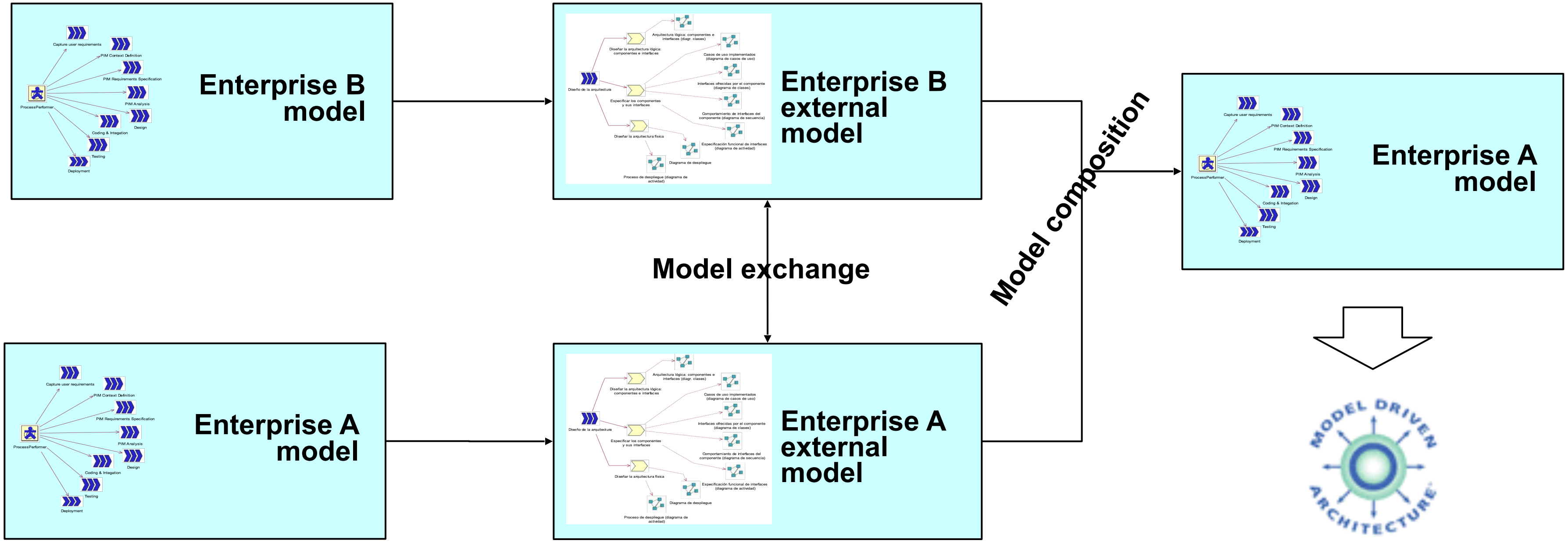
Domain profiles and models → E-commerce systems

Technical profiles and models → Real-time
QoS

Platform profiles and models → Objects
CORBA
Components
CCM/J2EE/.NET
Services
SOAP/WSDL/UDDI

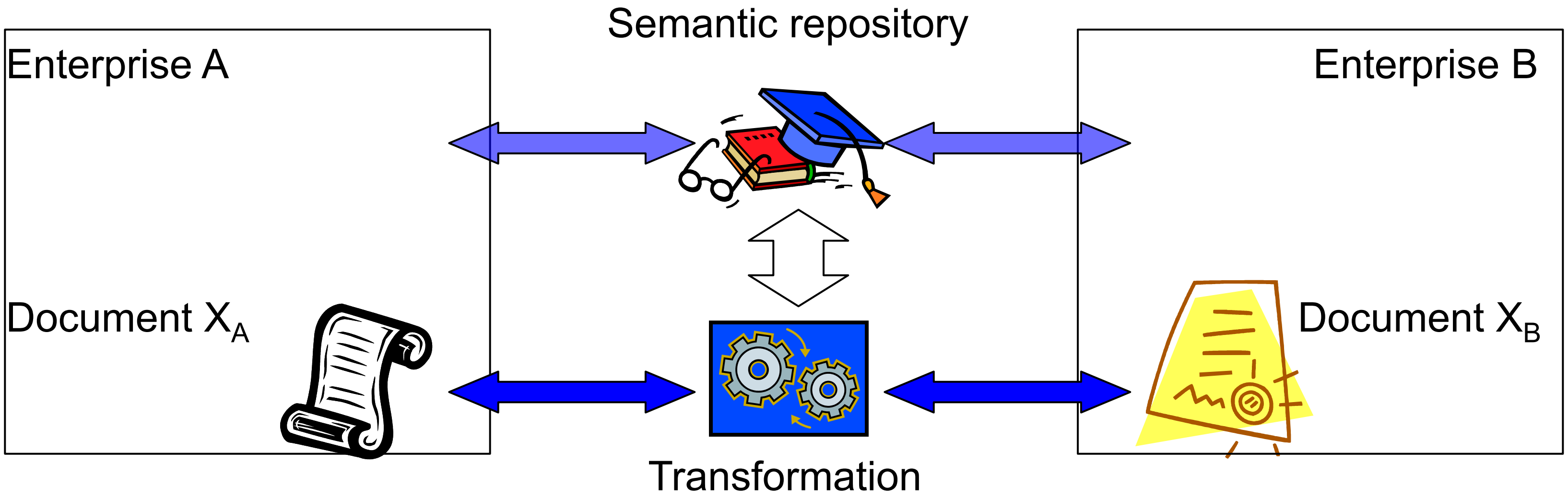
MDA and interoperability

- Interoperability from models point of view.
  - MDA approach tries to build a interoperable model, from enterprise models and processes to apply MDA mechanisms.

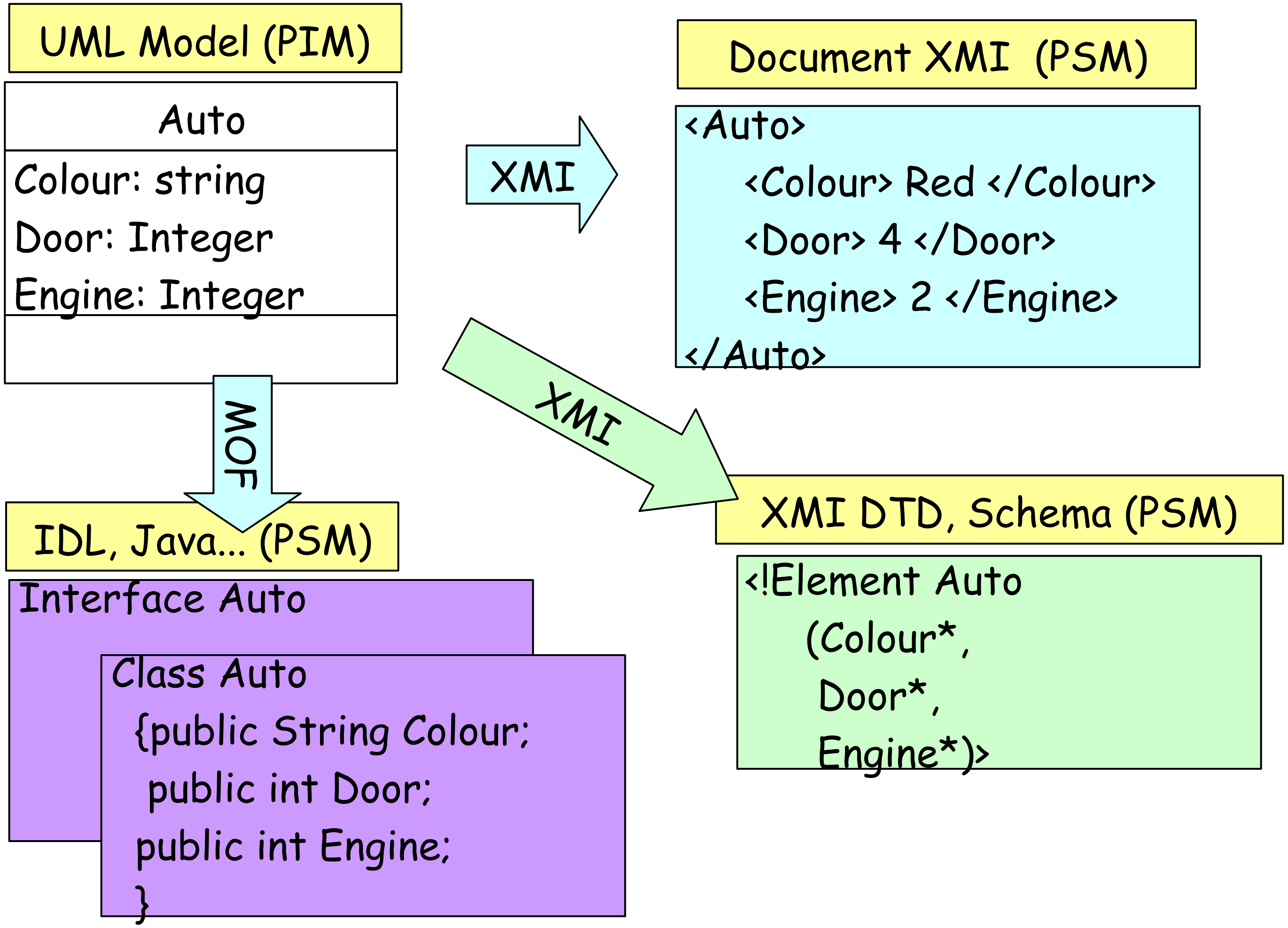

Enterprise B model
Enterprise B external model
Model exchange
Enterprise A model
Enterprise A external model
Model composition
Enterprise A model
MODEL DRIVEN ARCHITECTURE
Capture user requirements
PIM Context Definition
PIM Requirements Specification
PIM Analysis
Design
Coding & Integration
Testing
Deployment
ProcessPerformer
Arquitectura lógica: componentes e interfaces (diagr. clases)
Diseñar la arquitectura lógica: componentes e interfaces
Casos de uso implementados (diagrama de casos de uso)
Diseño de la arquitectura
Interfaces ofrecidas por el componente (diagrama de clases)
Especificar los componentes y sus interfaces
Comportamiento de interfaces del componente (diagrama de secuencia)
Diseñar la arquitectura física
Especificación funcional de interfaces (diagrama de actividad)
Diagrama de despliegue
Proceso de despliegue (diagrama de actividad)

MDA and interoperability

- Using transformations to get interoperability
  - It allows document transformations on the fly.
  - It can contribute to new approaches for semantic interpretations on information exchanges.



PIM to PSM transformations



UML Model (PIM)

Auto

Colour: string

Door: Integer

Engine: Integer

Document XMI (PSM)

```
<Auto>
    <Colour> Red </Colour>
    <Door> 4 </Door>
    <Engine> 2 </Engine>
</Auto>
```

IDL, Java... (PSM)

```
Interface Auto
```

```
Class Auto
 {public String Colour;
  public int Door;
 public int Engine;
 }
```

XMI DTD, Schema (PSM)

```
<!Element Auto
    (Colour*,
     Door*,
     Engine*)>
```

Transformation rules

UML model	XMI document	XMI DTD, schema	IDL, Java
class	<name.class> </name.class>	<!Element name.class()>	Interface name.class Class Auto{ }
attribute	<name.attribute> value </name. attribute>	name.attribute*	public datatype name.attribute;

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org..">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="model">
<xsl:apply-templates select="package"/> </xsl:template>
<xsl:template match="package">
```

The specification of a package transformation

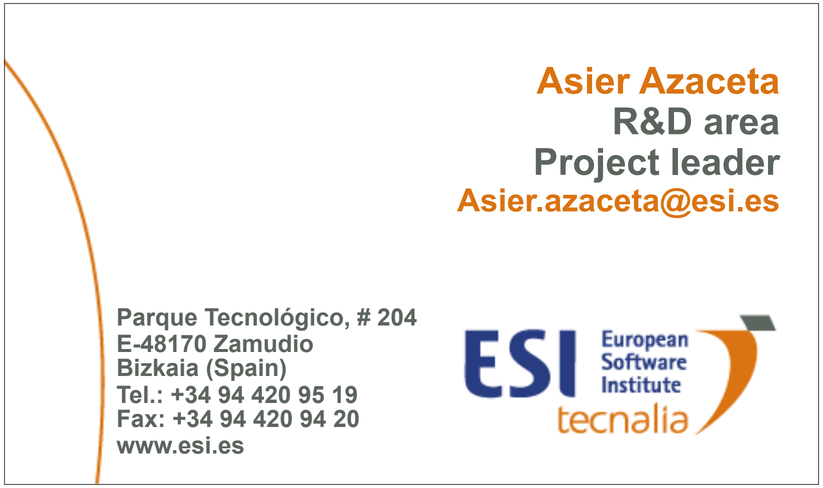
```
<xsl:apply-templates/> </xsl:template>
<!-- *** template match class -->
<xsl:template match="class">
```

The specification of a class transformation

```
<xsl:apply-templates select="attribute"/>
<xsl:apply-templates select="association"/>
<xsl:apply-templates select="operation"/> </xsl:template>
............
```

MDA references

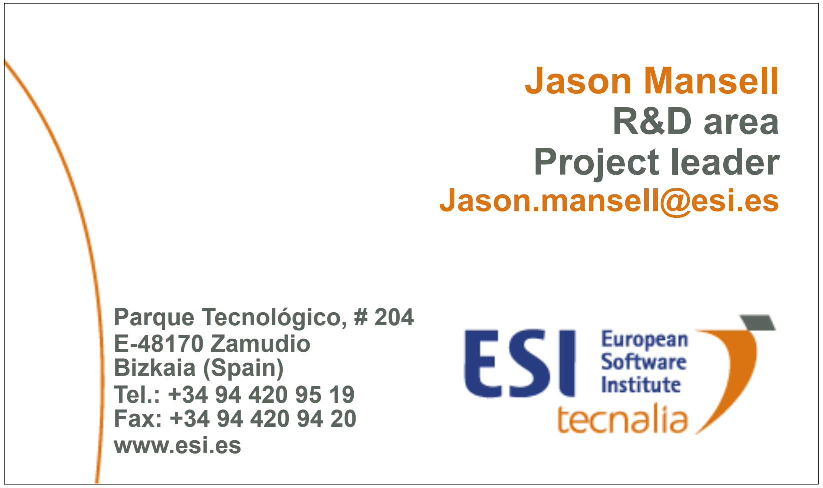
- The Object Management Group (OMG): http://www.omg.org
- MDA Guide: http://www.omg.org/mda/



Asier Azaceta
R&D area
Project leader
Asier.azaceta@esi.es

Parque Tecnológico, # 204
E-48170 Zamudio
Bizkaia (Spain)
Tel.: +34 94 420 95 19
Fax: +34 94 420 94 20
www.esi.es

ESI European Software Institute
tecnalia



Jason Mansell
R&D area
Project leader
Jason.mansell@esi.es

Parque Tecnológico, # 204
E-48170 Zamudio
Bizkaia (Spain)
Tel.: +34 94 420 95 19
Fax: +34 94 420 94 20
www.esi.es

ESI European Software Institute
tecnalia