A presentation of MDD basics

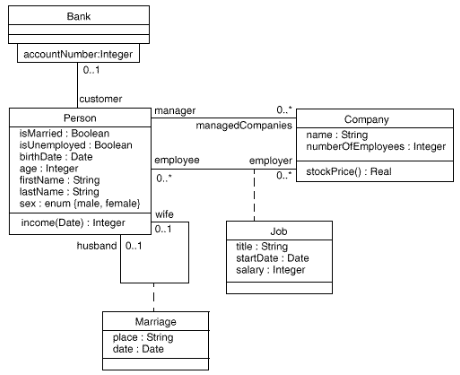
Model-driven development (MDD) tutorial for managers

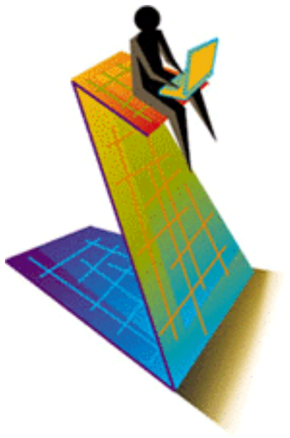
EUROPEAN SOFTWARE INSTITUTE,
Corporación Tecnológica Tecnalia
Parque Tecnológico, # 204
E-48170 Zamudio
Bizkaia (Spain)
www.esi.es

Context of this work

- The present courseware has been elaborated in the context of ModelWare European IST FP6 project (http://www.modelware-ist.org/)
- The MODELWARE project (Modelling solution for software systems) brings together 19 partners from Europe and Israel. Its main objectives are to develop a solution to reduce the cost of software systems large-scale deployment by the means of Model Driven Development techniques.
- To achieve the goals of large-scale dissemination of MDD techniques, ModelWare is also promoting the idea of collaborative development of courseware in this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

Contents


Bank
accountNumber:Integer
0..1
customer
Person
isMarried : Boolean
isUnemployed : Boolean
birthDate : Date
age : Integer
firstName : String
lastName : String
sex : enum {male, female}
income(Date) : Integer
manager
0..*
managedCompanies
Company
name : String
numberOfEmployees : Integer
stockPrice() : Real
employee
employer
0..*
0..*
wife
0..1
husband
0..1
Job
title : String
startDate : Date
salary : Integer
Marriage
place : String
date : Date

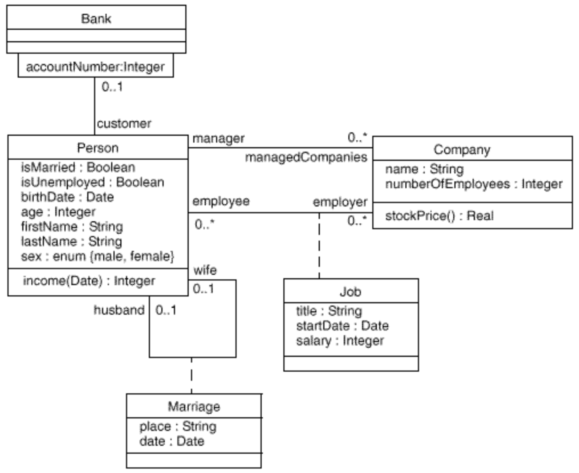


Model Driven Development (MDD)

- Software is getting **easier to develop** thanks to languages, tools and processes, but getting from the user requirements to the final solution is **still difficult.**

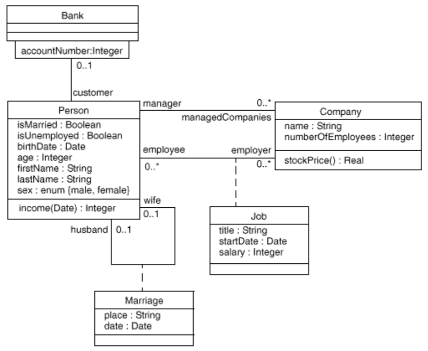
- **Complexity** is an ever-raising property of software systems
  - **Abstraction** and subjects separation help **managing complexity**
    - Abstraction allows to concentrate in what's important
    - Abstraction may remove important implementation details
  - in order to **increase the productivity** we need
    - Automation
    - Reuse
    - Capitalisation of designs
- **Modelling** is a natural way for doing this.
  - A **model** is an abstract representation or a simplification of something.

- **Focus on business not code !**



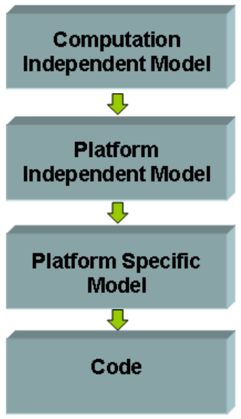
What is MDA? (I)

- Just like UML, MDA is a standard promoted by the **OMG.**
- *A* **set of specifications** defined by OMG's open, worldwide process.
- Model-Driven Architecture (MDA) is a new way to look at software development, from the point of view of the **models**.
- **Models** are the core; **Design** is the focus.
- MDA supports **technology-independent design.**
- MDA divides **domain** knowledge and **platform** knowledge.



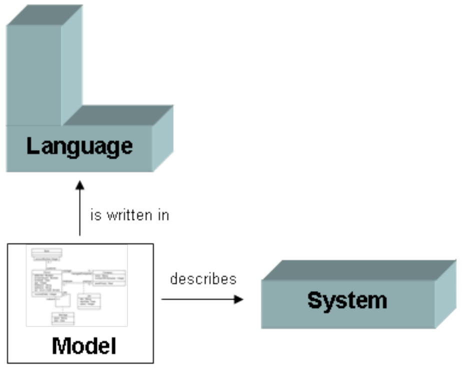
What is MDA? (II)

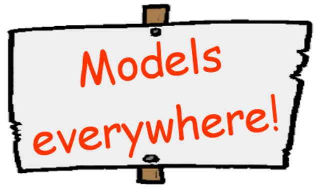
- **Separates** the operational specification of a system from the details such as how the system uses the platform on which it is developed.
- MDA provides the means to:
  - **Specify** a system independently of its platform
  - Specify platforms
  - Choose a platform for the system
  - **Transform** the system specifications into a platform dependent system
- Three fundamental **objectives**:
  - Portability
  - Interoperability
  - Reuse
  - Productivity (derived objective)

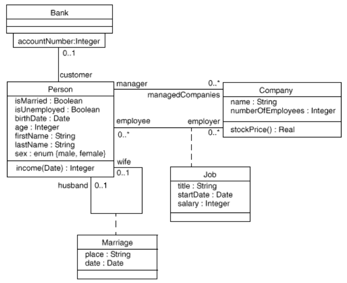


MDA Basic Elements: Models (I)

- **Cornerstone** of MDA
- **Abstraction** of a reality, different from it, and that can be used for (re)producing such reality.
- Expressed in a **well-defined language** (syntax and semantics) which is suitable for automated interpretation.
- In MDA, "**everything is a model**"
- One model may describe only part of the complete system.
- A model helps
  - Focusing on essentials of a problem to better understand it.
  - Moving towards an effective solution.



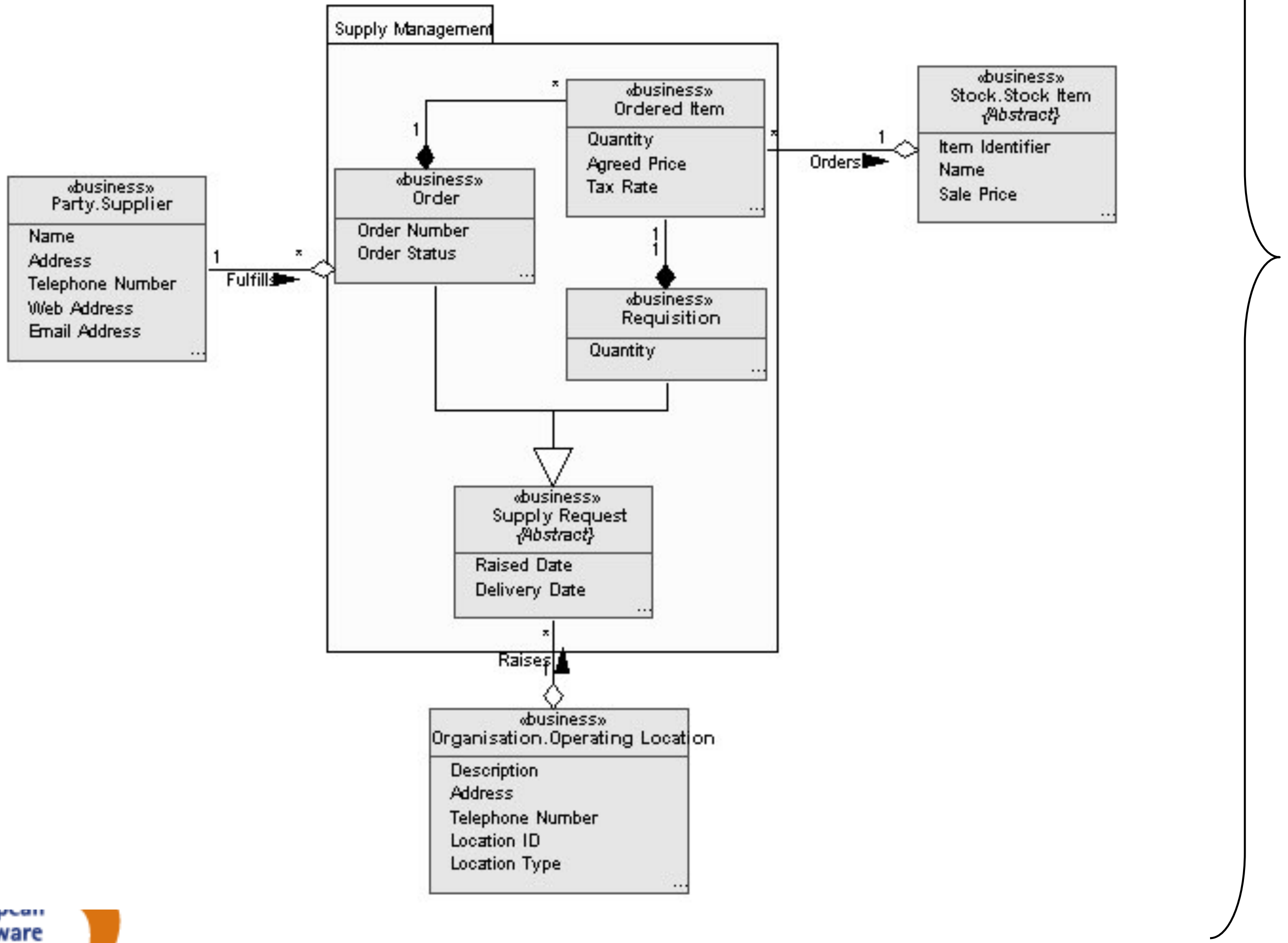




MDA Basic Elements: Models (II)

- Types of models:
  - **Business models** or Computation Independent Models (**CIM**)
    - Defines the domain identifying fundamental **business entity** types and the relationships between them
    - Say **nothing** about the software systems used within the company.

MDA Basic Elements: Models (III)



Computation Independent Model

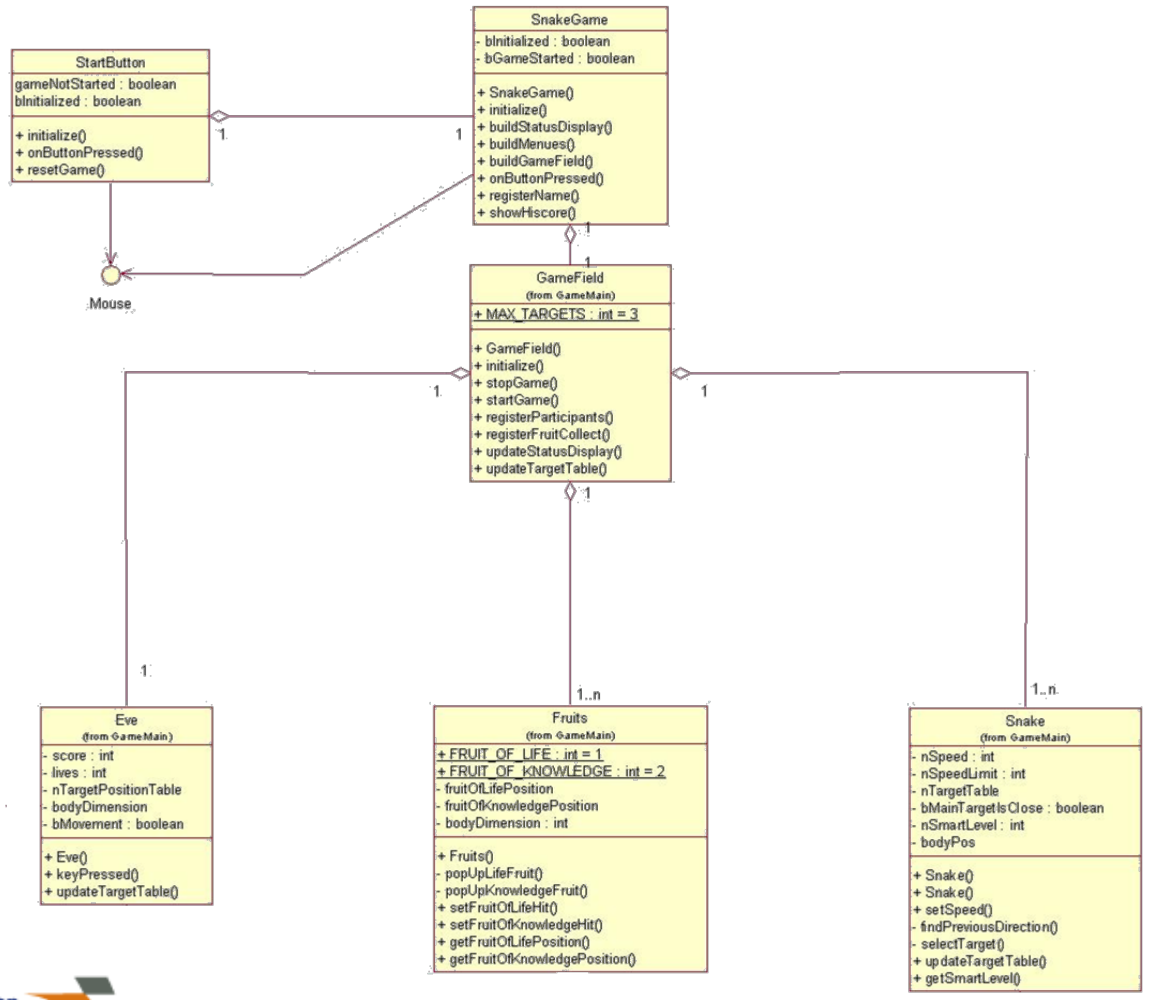
Platform Independent Model

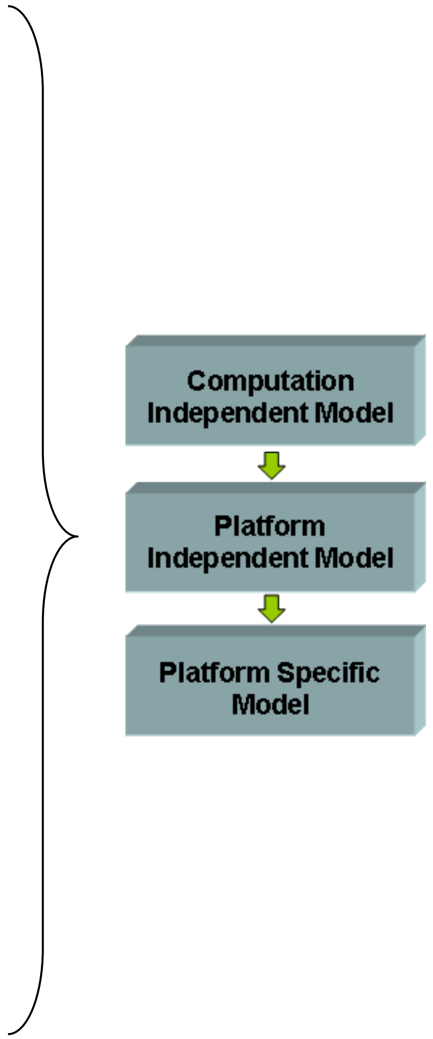
Platform Specific Model

MDA Basic Elements: Models (IV)

- Types of models:
  - **Business models** or Computation Independent Models (**CIM**)
    - Defines the domain identifying fundamental **business entity** types and the relationships between them
    - Say **nothing** about the software systems used within the company.
  - **System models**
    - These models are a description of the software system.
    - Platform **independent** models (**PIM**):
      - Resolves functional requirements through purely problem-space terms.
      - **No platform-specific details** are necessary.

MDA Basic Elements: Models (V)

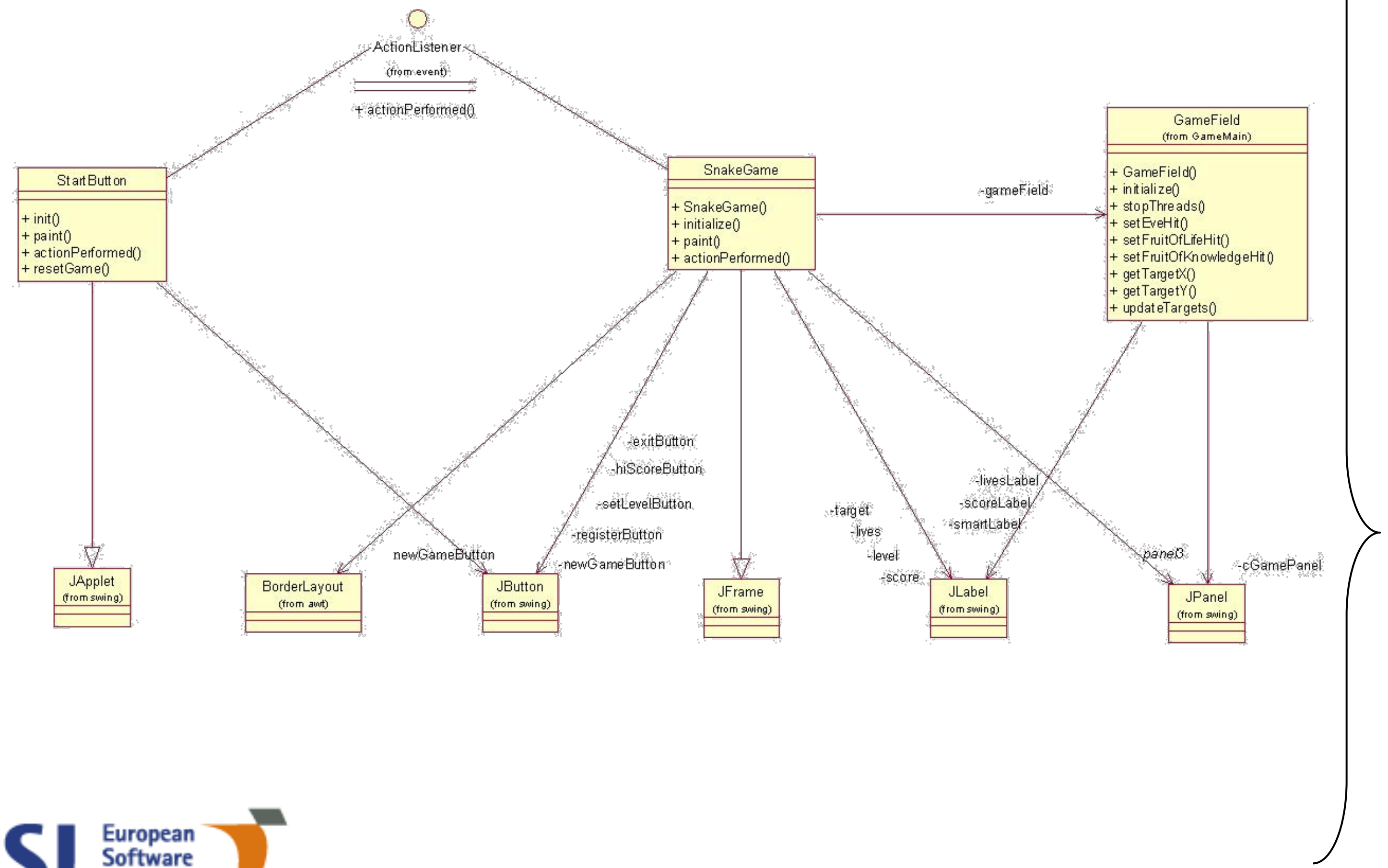


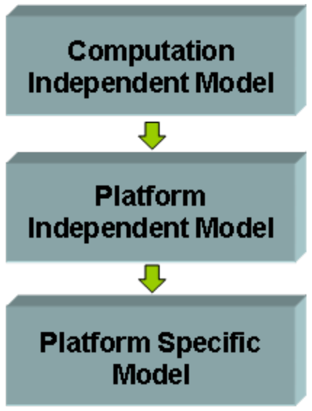


MDA Basic Elements: Models (VI)

- Types of models:
  - **Business models** or Computation Independent Models (**CIM**)
    - Defines the domain identifying fundamental **business entity** types and the relationships between them
    - Say **nothing** about the software systems used within the company.
  - **System models**
    - These models are a description of the software system.
    - Platform **independent** models (**PIM**):
      - Resolves functional requirements through purely problem-space terms.
      - **No platform-specific details** are necessary.
    - Platform **specific** models (**PSM**):
      - It is a **solution model** that resolves both functional and non-functional requirements.
      - **Requires information on specific platform** related concepts and technologies.
    - Platform independence is a **relative term**.

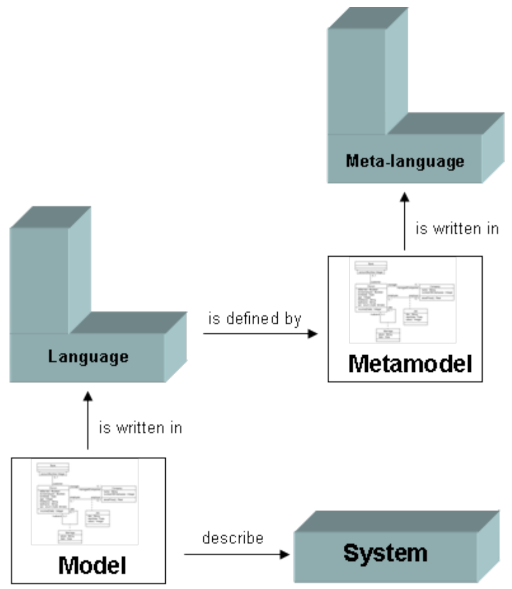
MDA Basic Elements: Models (VII)

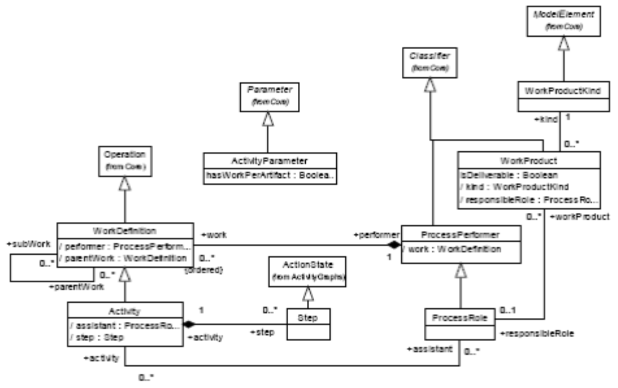




MDA Basic Elements: Metamodels (I)

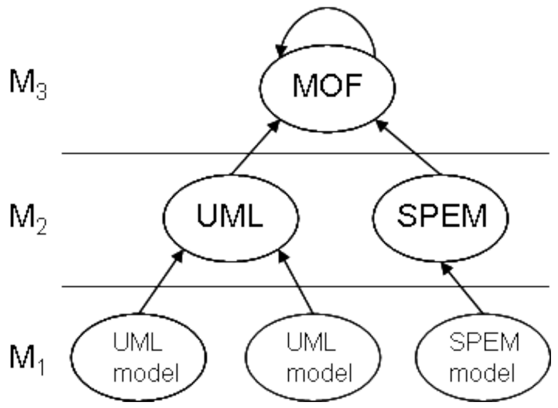
- Allows the exchange of models among modelling tools.
- Allows the representation of a specific domain elements
  - Use of a common terminology.
  - Reduce misunderstandings
  - Production of a complete documentation
  - Check of consistent processes
  - Traceability of process artefacts: impact analysis
- A metamodel:
  - **Is also a model** and must be written in a well-defined language.
  - Defines **structure**, **semantics** and **constraints** for a family of models.



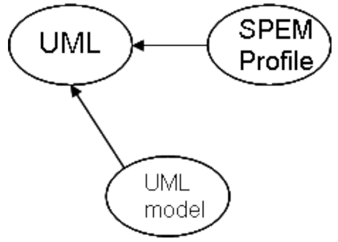


MDA Basic Elements: Metamodels (II)

- The three-layer architecture:
  - (M3) Metametamodel:
    - **One** unique meta-meta-model, the **Meta-Object Facility (MOF)**.
    - It is some kind of "**top level ontology**"
  - (M2) Metamodel:
    - Defines **structure**, **semantics** and **constraints** for a family of models.
  - (M1) Model:
    - Each of the models are **defined** in the language of its **unique metamodel**.

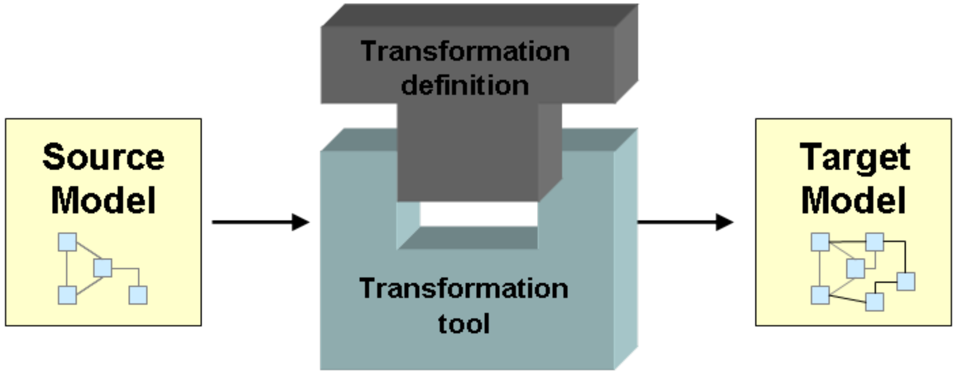


- UML profiles are **adapted modelling languages**.

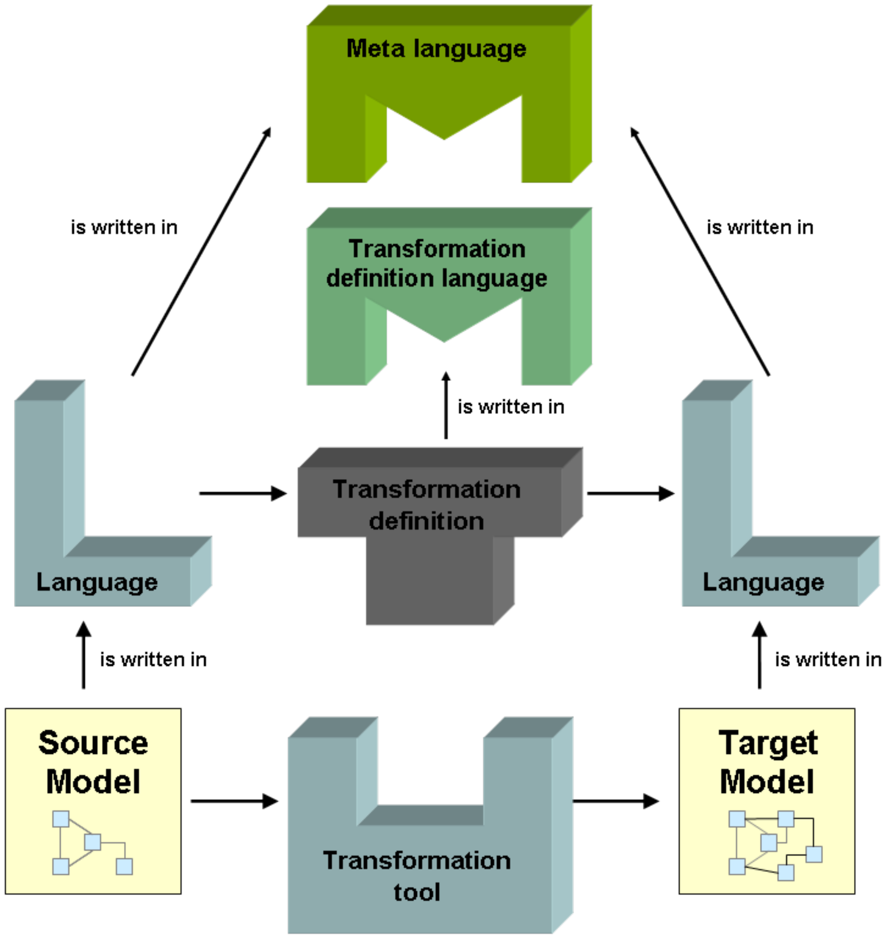


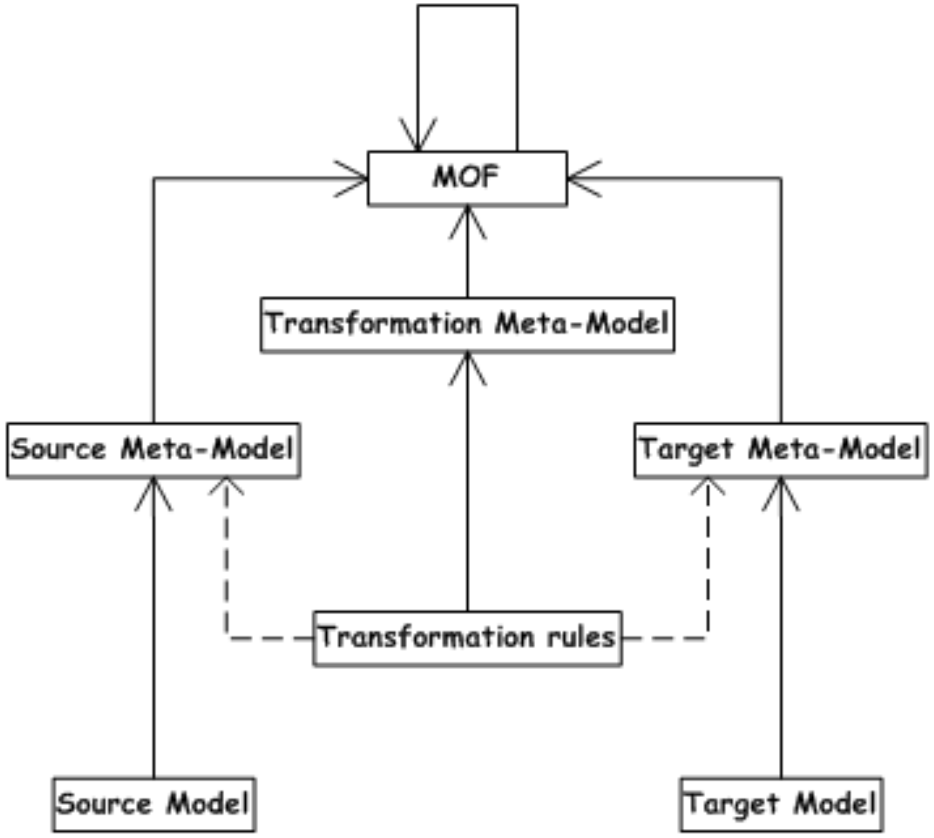
MDA Basic Elements: Transformations (I)

- A **transformation** is the automatic generation of a **target** model from a **source** model, according to a **transformation definition**.
- A **transformation definition** is a set of **transformation rules** that together describe how a model in the source language can be transformed into a model in the target language.
- A **transformation rule** is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language.



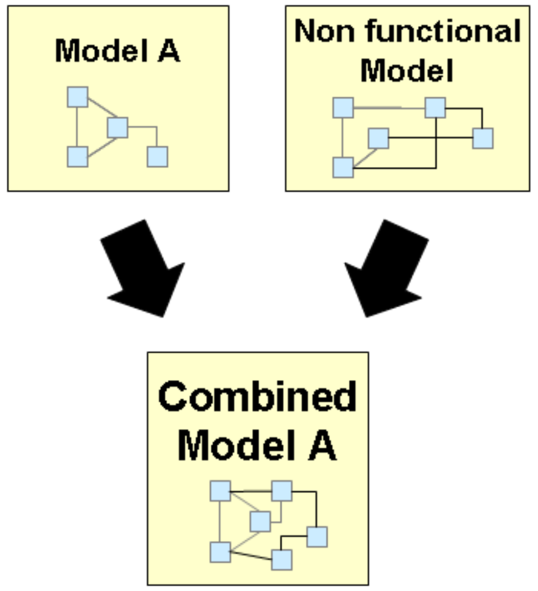
MDA Basic Elements: Transformations (II)

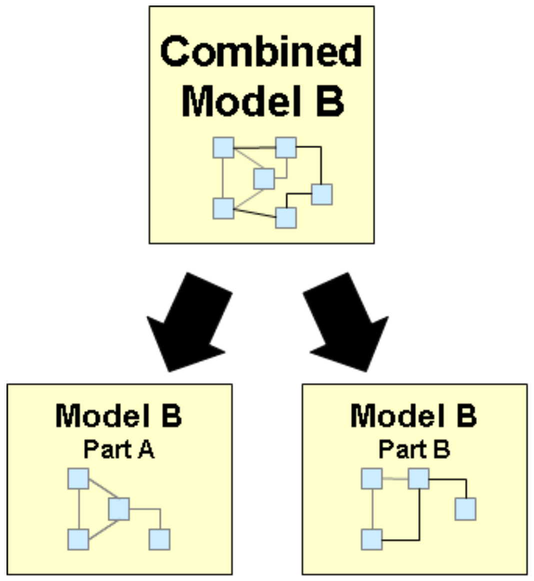




MDA Basic Elements: Transformations (III)

- Composition:
  - It is a special case of transformation.
  - Allows bringing new details or “aspects” into a model.
  - Allows splitting functionality across several platforms



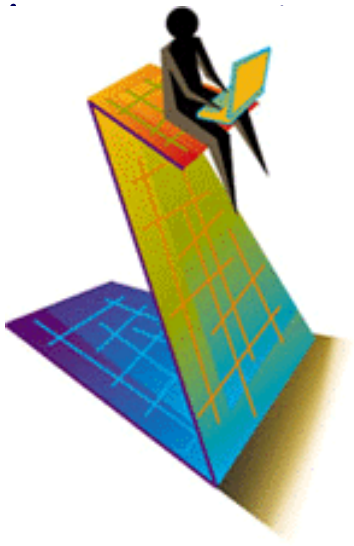


MDA Technologies and Standards

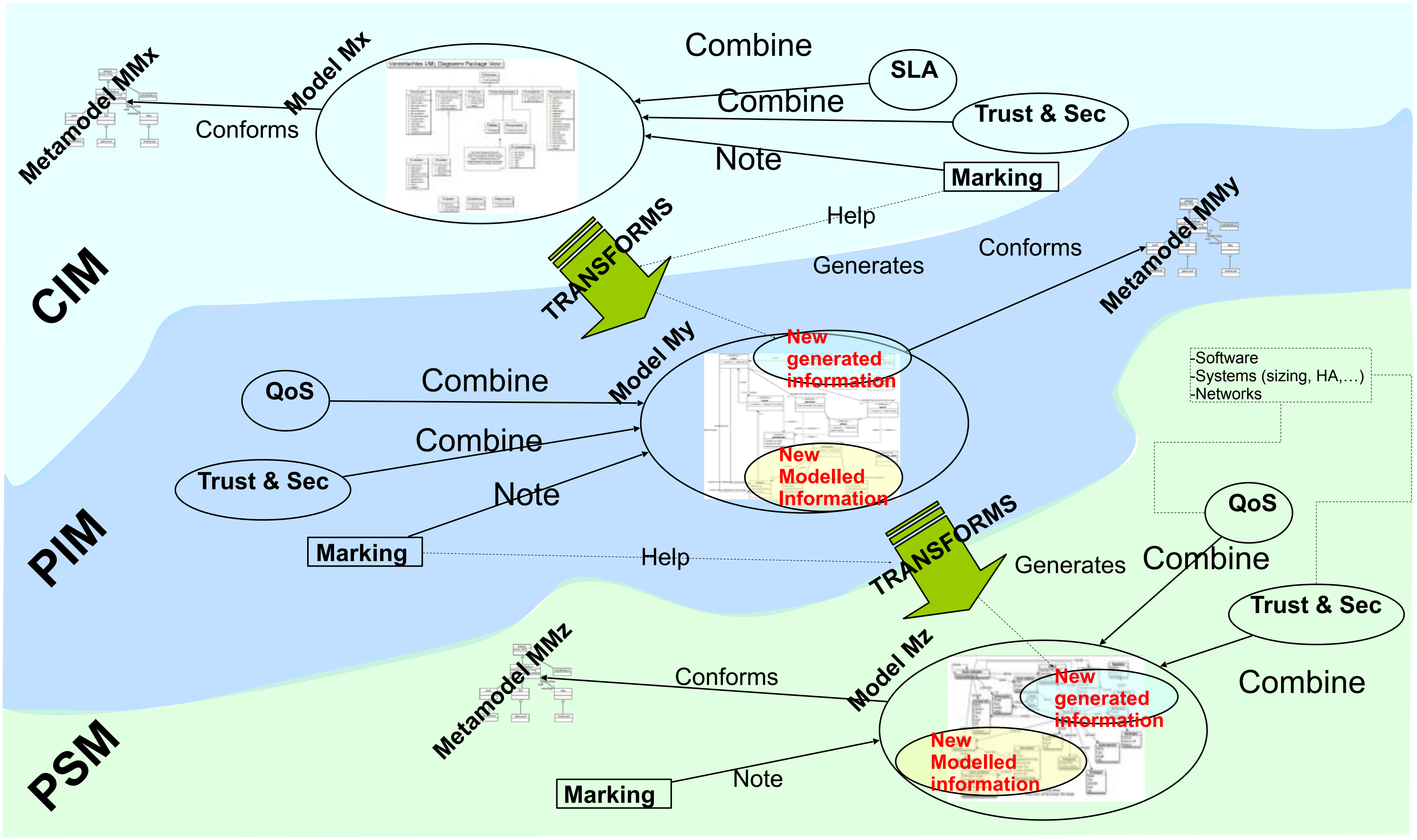
- **MOF**: Meta-modelling language, repository interface (JMI), interchange (XMI)
- **UML**: Standard modelling language. Instance of the MOF model. For developers and “meta-developers”
- **CWM**: Modelling languages for data warehousing applications. (e.g. Relational DBs)
- **OCL**: expression language, extends the expressive power of UML and MOF
- **QVT**: Transformations definition language. Also for Queries and Views of models.
- **SPEM**: metamodel and a UML profile used to describe a concrete software development process.

MDA Development Process (I)

- A change in the approach to development requires a **change in the processes** and methodologies. New activities and work products will appear.
- It should not be a complete change to our way of working but an adoption of the technology, **adapting** our way of working
- **Methodological tools** are required for still keeping projects monitored and under control
- Some steps of the development are **no more** **verhead** but "real" developn ough MDA



MDA Development Process (II)



Climbing MDA mountain (I)

- Benefits of MDD:
  - **Flexible** implementation: platform changes
  - Simpler and more effective **maintenance**
  - **Effective** development: Common language; Requirements **traceability**; Earlier testing and **simulation**
  - Increased **productivity**: Automation; Increases reuse; Reduction of rework
  - **Quality** improvement.
  - Updated **documentation** of the system.
  - Ensures customers, designers and architects **understanding**.

Climbing MDA mountain (II)

- Difficulties of adopting MDD:
  - Shift in development **culture**. **Staff** not ready for modelling. New roles are needed.
  - Difficult to distinguish **real MDA** providers
  - Lack of **confidence** on MDA promises being real
  - Usually seen as a **heavyweight** methodology
  - **Transformations** promises not a reality yet.
  - Incomplete and not interoperable nor integrated **Tool chain**.
  - Relatively **high cost of adoption** (training, infrastructure, tools)
  - Definition of an extension mechanism to allow customization and specialization without breaking the code generation

Marketing MDA vs. Real MDA (I)

"You build a **platform-independent** model of your application **with UML®**, and from that model you can **automatically generate platform-specific models and code** for a variety of target platforms."

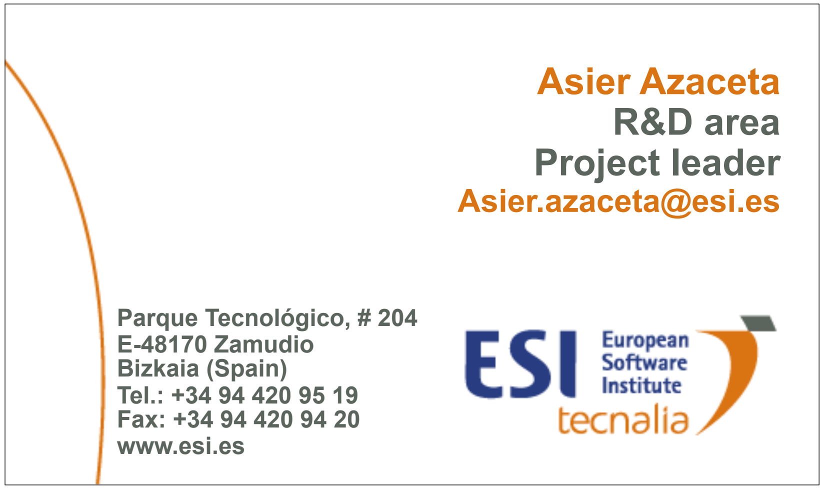
- **Flexible implementation:**
  - ⊖ Platform independence is a **relative** term.
  - ⊖ Strong dependence on **quality** of models and transformations.
  - ⊖ High importance of **maintaining** the modelling approach (instead of tweaking the code).
  - ⊕ **Derivation** of different PSM is possible.
  - ⊕ **Separation** of concerns: allow stakeholders to be focused on a specific domain.

Marketing MDA vs. Real MDA (II)

- **Flexible and easier integration:**
  - ➕ Two different focuses: data integration and functionality integration.
  - ➕ Model (conceptual) integration is easier than application integration
- **Increased productivity:**
  - ➖ Requires people to be trained in modelling -> analysts vs. programmers
  - ➖ Requires the development of basic "infrastructure"
  - ➕ Automates steps of the development process
  - ➕ Reduces the amount of rework due to errors
  - ➕ Reduces the loss of information from logical to technical implementation

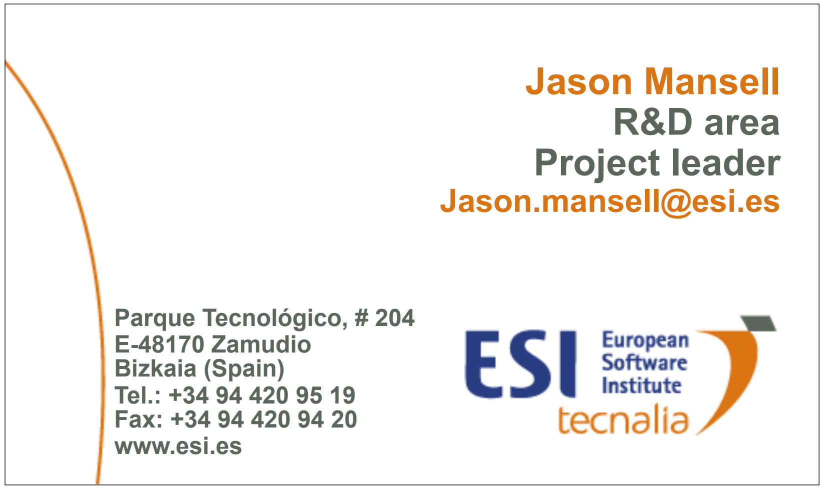
Marketing MDA vs. Real MDA (III)

- **Effective Development:**
  - ⊕ Improves requirements traceability: changes and validation
  - ⊕ Facilitates early testing and simulation
- **Simpler and more effective maintenance:**
  - ⊕ Documentation exists for developed applications
  - ⊕ Changes can be done directly to existing designs


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