Model Driven Software Engineering Environment

Xavier Blanc – INRIA Lille - EPI Adam

Outline

- Origin of MDA
- MDA Results
- Toward a MDSEE
- Experiences

In 2000

- CORBA was a powerful first step, but we have more steps to take.
- Over the past decade or more, companies have endured a succession of middleware platforms.
- Companies that adopt the MDA gain the ultimate in flexibility: the ability to derive code from a stable model as the underlying infrastructure shifts over time.
- ROI flows from the reuse of application and domain models across the software lifespan.

The 3 Goals of MDA

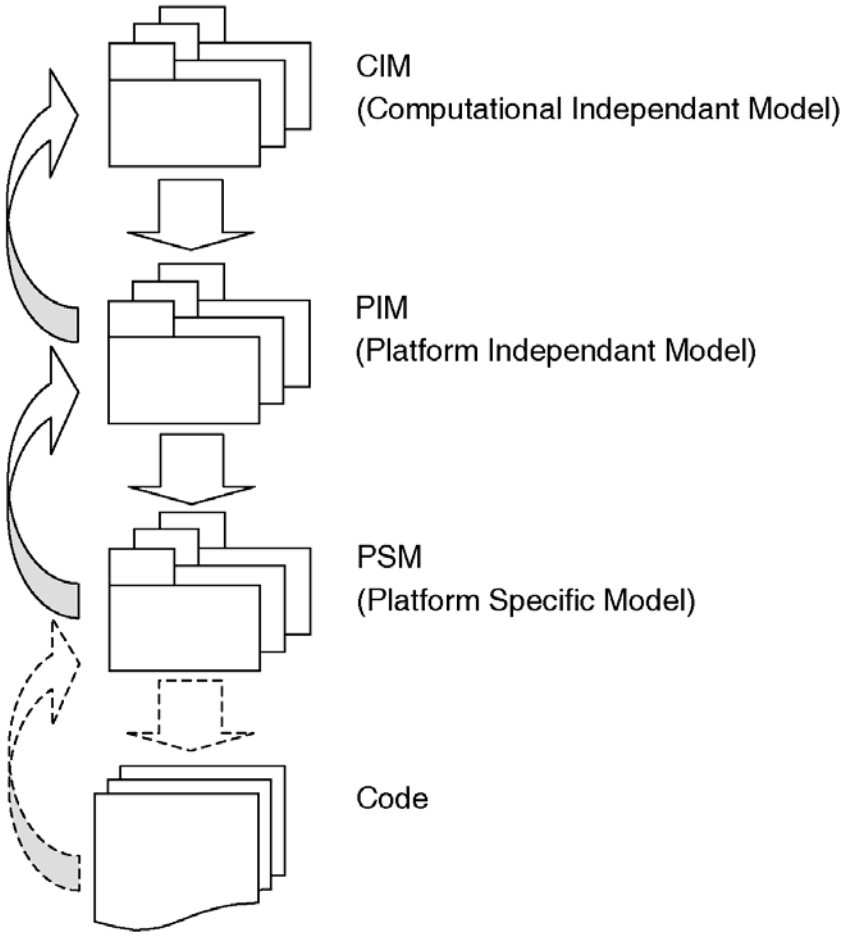
- **Durability**: models should be stable (more than the code)
- **Productivity**: models should be used to generate (all) the code
- **Platform**: models should be platform independent (as possible)

Main challenge

Interoperability

- How to create stable application and domain models ?
- How to reuse them for building new software systems or maintaining existing ones ?

MDA Approach

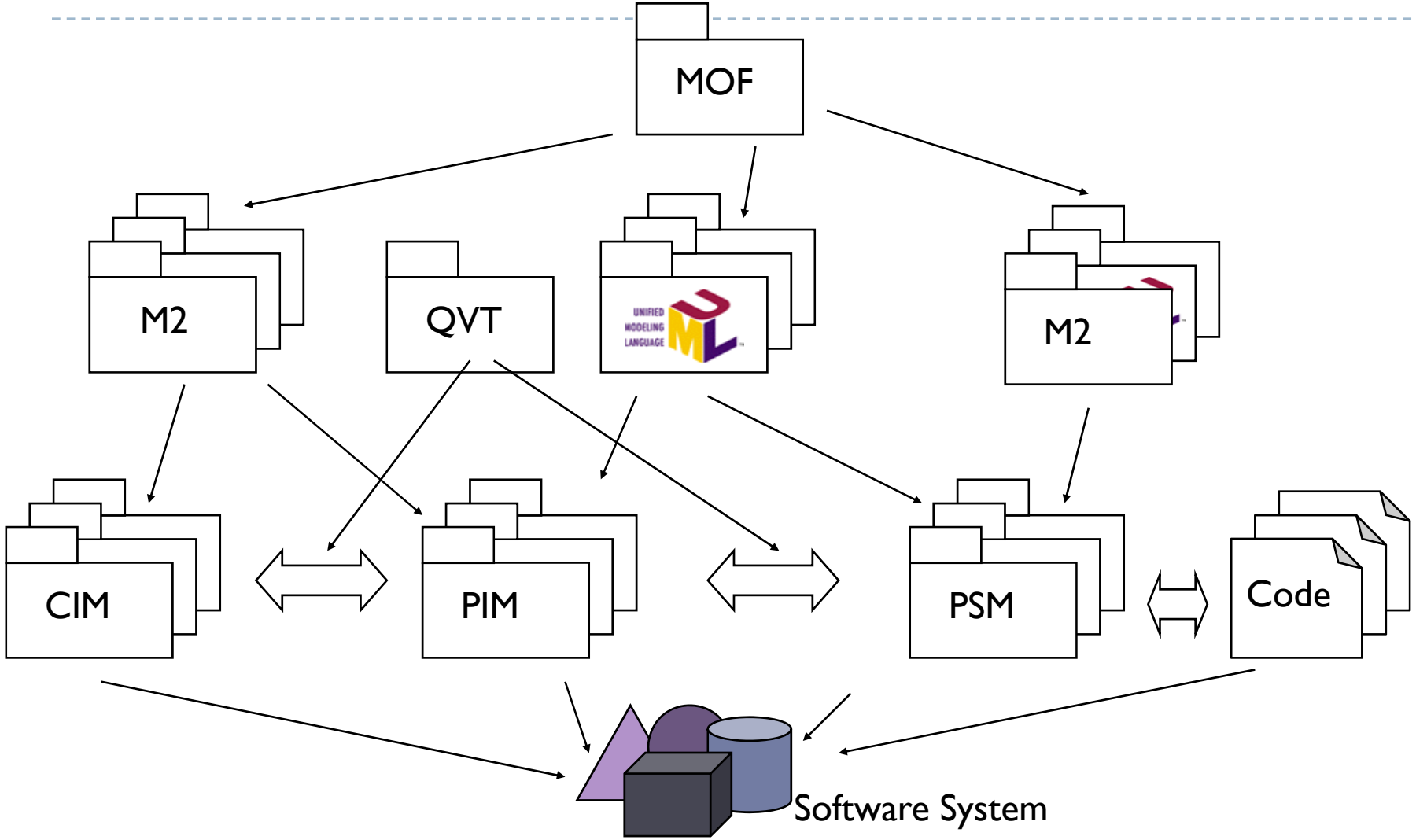


Requirements

Architecture

Design

MDA Architecture



Durability (1/2)

The biggest success of MDA

- Success of the 3 layers architecture => EMF
- No more meta-meta-model wars
- Models everywhere

Durability (2/2)

Wrong Truths

- CIM, PIM, PSM, Code
  - Not a real methodology, no reuse, traceability nightmare
- Mn+1 stable implies Mn stable
  - Meta-models and models do evolve
- Model = Formal
  - What is a model ? What is a meta-model ? What means conformance ?

Productivity (1/2)

Rich domain

- Code generators
  - 100% of the code can be generated
- Model Transformation
  - (too?) many languages
- Success stories and industry investment
  - "Y" Cycle

Productivity (2/2)

Holly Grail ?

- CIM 2 PIM 2 PSM 2 Code
  - From intention to realization !
- Code generation
  - Currently are either marketing or very specific
- Model Transformation
  - In progress? (Mono-directional, binary, not incremental)
- How to make a productive model ?
  - Everything is in the model!
- Model semantics ?

Platform

Not so many results

- What platform means ?
  - Platform ontology
- Where to put the cursor?
  - The more abstract, the more useless
- Platform innovation!

Personal Analysis

- Existing useful technologies
  - Meta-model, model, model transformation, model verification, model simulation, …
- Current challenges
  - Semantics/ Formal Foundation, relation between models
- Too rigid
  - method, meta-model, transformation, static process

A new? deal

Living models

- Models are living artifacts used in several projects.
- Meta-models, model transformations, model simulations, model verifications are living artifacts too.

How to engineer their life cycles and their relationships?

MDSEE

- MDSEE should support the MD developers during their project life cycle
  - Model access (read, write,)
  - Meta-model access (verify conformity, instantiate)
  - Model transformation (read, write, run)
  - Model simulation (read, write, run)
  - Process (read, write, run)
  - Project (create)

- MDSEE should support multi-model life cycle!

MDSEE Challenges

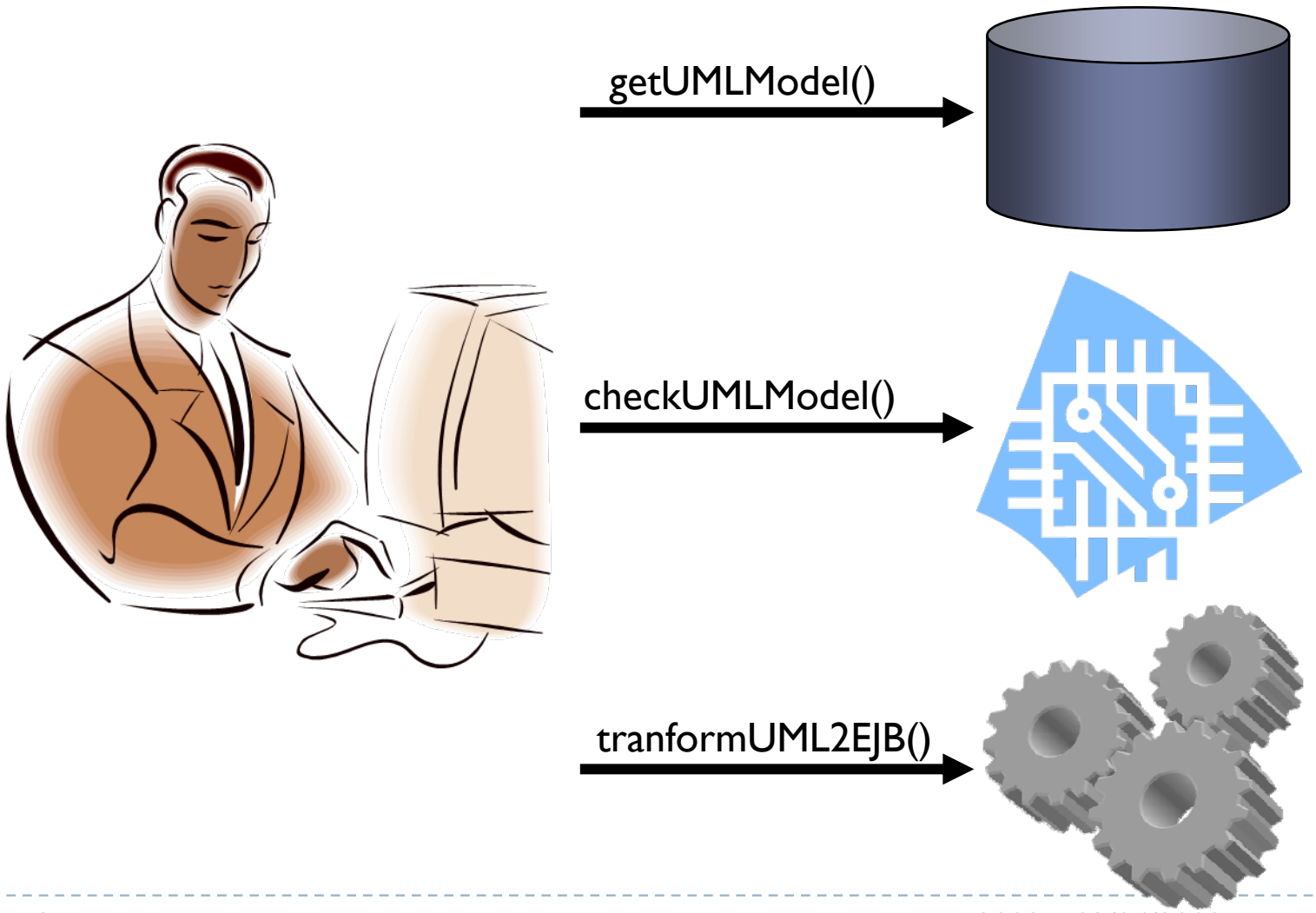
- How to provide model access?
- How to control model evolution?
- How to run model?
- How to control model runtime?
- How to support developers' interactions?

- How to integrate legacy environment ?

ModelBus (1/3)

- First attempt to build MDSEE
- Modeling Service Oriented
  - Model Accesses are MS
  - Transformations are MS
  - Simulation, Verification are MS
- Validation
  - 2 European Projects
  - 10 tools plugged as MS
  - Scenario with choreography

ModelBus (2/3)



ModelBus (3/3)

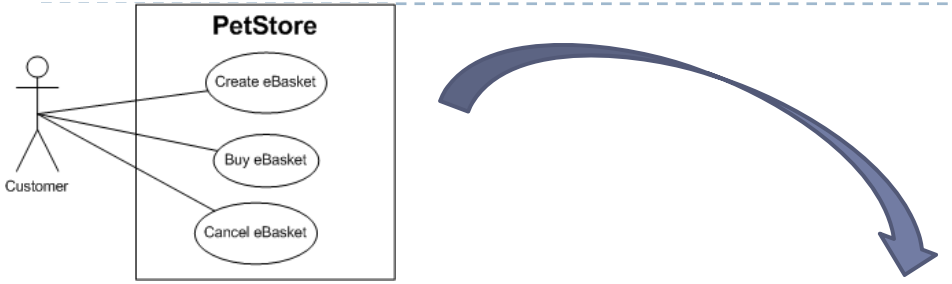
- Who is responsible for MS ?
  - Trust ? Localization (where are the models)?
- MS are stateless
  - No model life cycle support
- No support for project
- No support for process

- Difficulty for integrating existing tools

Praxis (1/3)

- Models are represented by sequences of operations
- Models are living entities
- Models life cycle can be controlled (consistency management)
- Sequences can be shared across sites (peer-to-peer)

Praxis (2/3)




```
01. create(c1,Class)
02. setProperty(c1,name, {'PetStore'})
03. create(uc1,UseCase)
04. setProperty(uc1,name, {'Buy eBasket'}))
05. create(uc2,UseCase)
06. setProperty(uc2,name,{'Create eBasket'})
07. create(uc3,UseCase)
08. setProperty(uc3,name,{'Cancel eBasket'})
09. setReference(c1,ownedUseCase,{uc1,uc2,uc3})
10. create(a1,Actor)
11. setProperty(a1,name, {'Customer'})
12. setReference(a1, usecase, {uc1,uc2,uc3})
```

Praxis (3/3)

- How to run models and control their runtime?
- How to support developers' interactions?
- How to integrate existing tools?

- **Power of operations!**

Conclusion

- Durability, Productivity, Platform
  - Abstraction is not enough
  - Multiple developers and projects
  - Everything does evolve

- **Living Models**

- Need of an MDSEE
  - Support of Models Life Cycle
  - Support of **multi-**(models, meta-models, views, projects, developers, …)