ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

On Metamodelling

PhD studentship admittance examination

June 21, 2004

Frédéric Fondement

Contents

- Software engineering
- Modelling
- Metamodelling
- A case study
- Concluding remarks

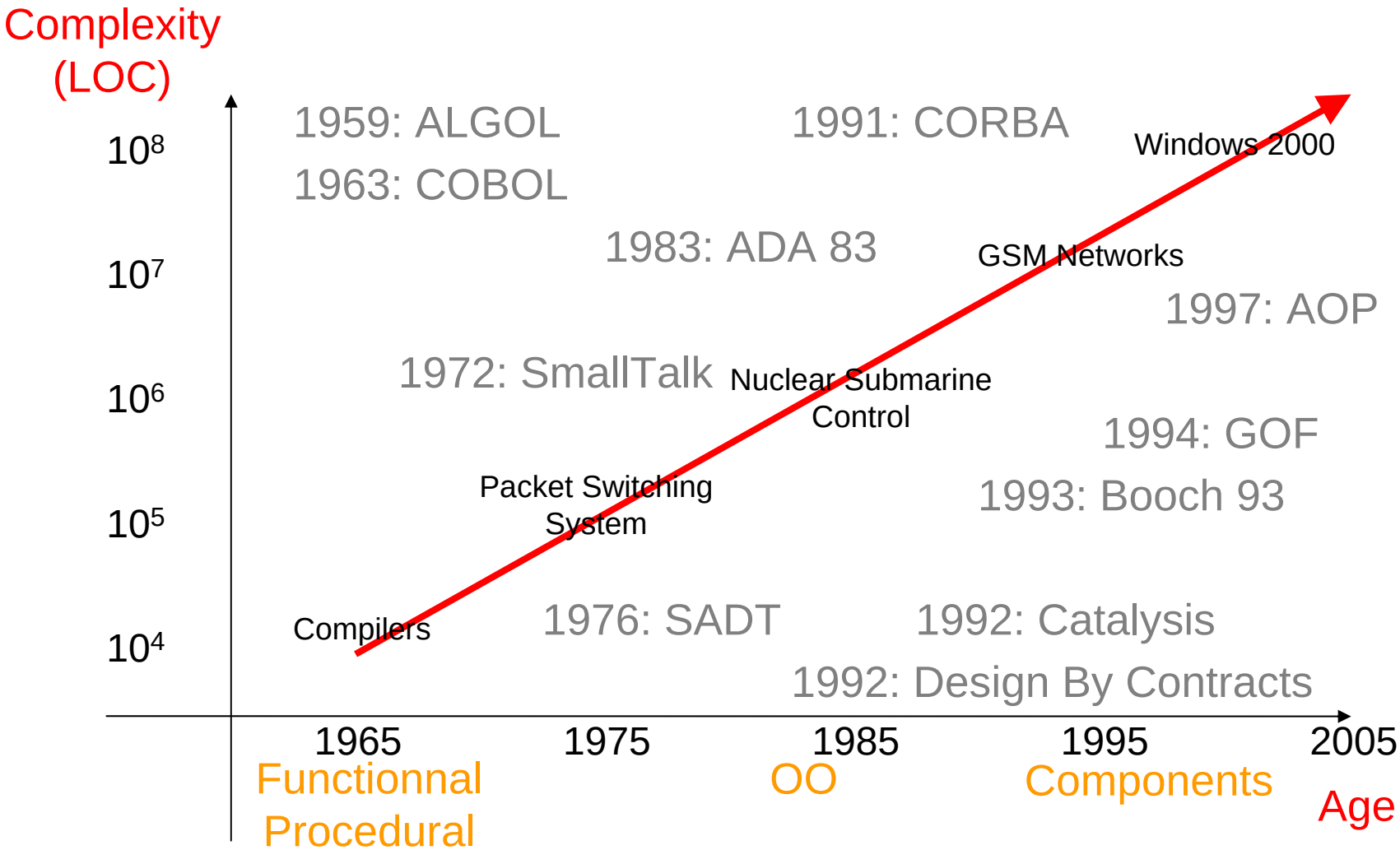
Contents

Productivity Gains in SE

- ***Methodologies***
  - SADT
  - Fusion
  - OMT
  - Booch
  - Catalysis
  - RUP
  - Fondue
  - SEAM
  - …

- ***Abstraction Techniques***
  - Punched Cards
  - Assembly Code
  - Functional / Procedural Programming
  - Object-Oriented Programming
  - Patterns
  - Concurrent Programming
  - Component-Oriented Programming / Middleware
  - Design by Contracts
  - Aspect-Oriented Programming
  - Product Family Engineering
  - …

Consequence: Production Gains



Remark: A new paradigm needs around 10 years to be mature

Major Lessons

- SE Best Practices
  - Reuse / Develop for reuse
  - Refinement / Refactoring
  - Prototyping
  - Test / Verification
  - Communication / Documentation
- Problems
  - Requirements ***change***
    - Nokia: 50% changed after finalization ; 60% of them at least twice
  - Platforms ***change***
    - OS, Languages, Databases, Middleware, …
  - Development technology ***change***
    - Compilers, Code generators, Frameworks, …
  - People ***change***
    - Especially for systems developed 30 years ago !

Contents

Models

- Abstract away details
  - Use paradigms rather than technologies
  - Enhance productivity and production !
- Human friendly
  - Different views of the same system
- Cheap to manipulate and maintain
- Models at any abstraction level
  - Use cases, Business models, SDL, CCM, B, …

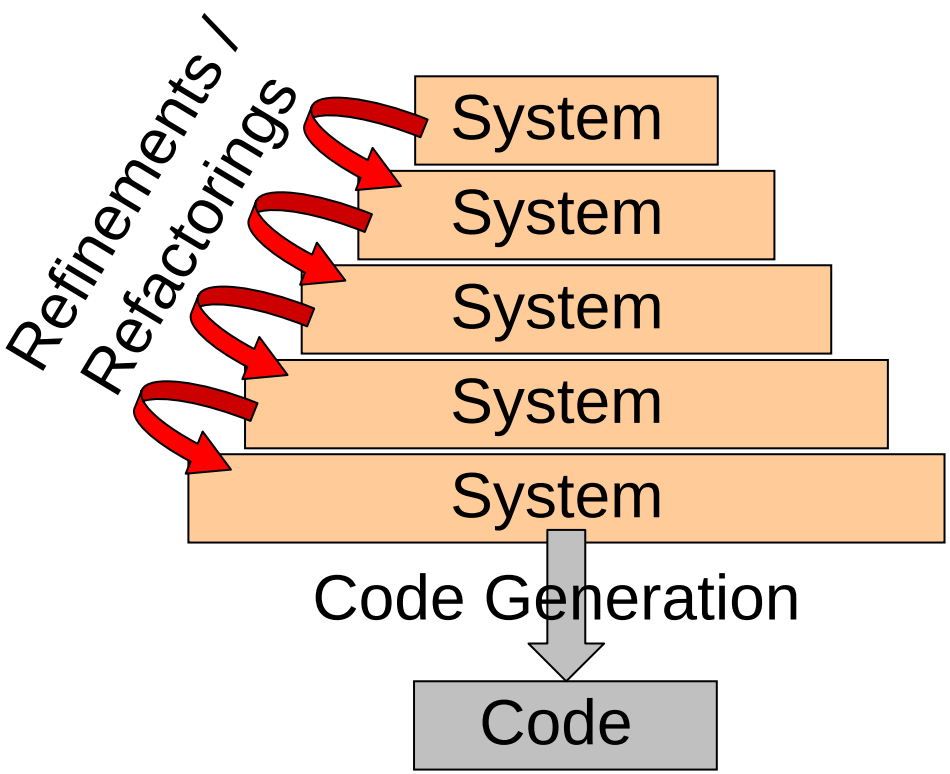
Used most of the time for

- Documentation / communication
- Analysis
- Prototyping

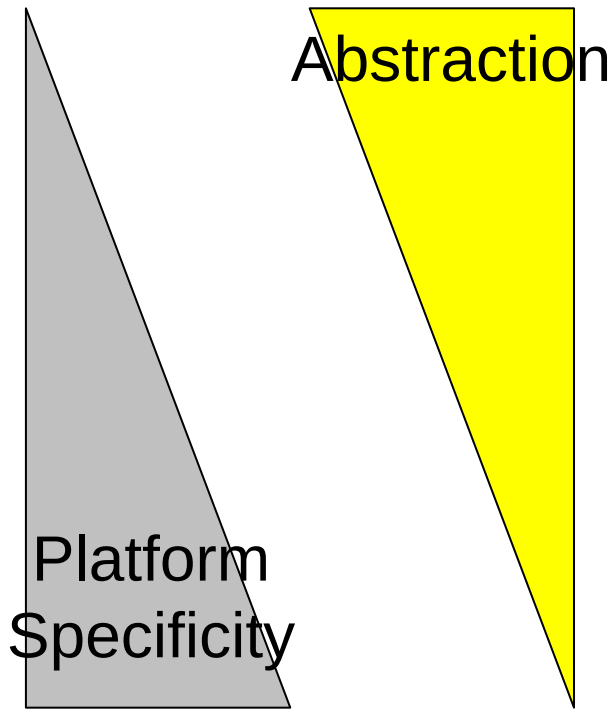
Often drawings out of sync with (code) reality !

SE with Model Refinements

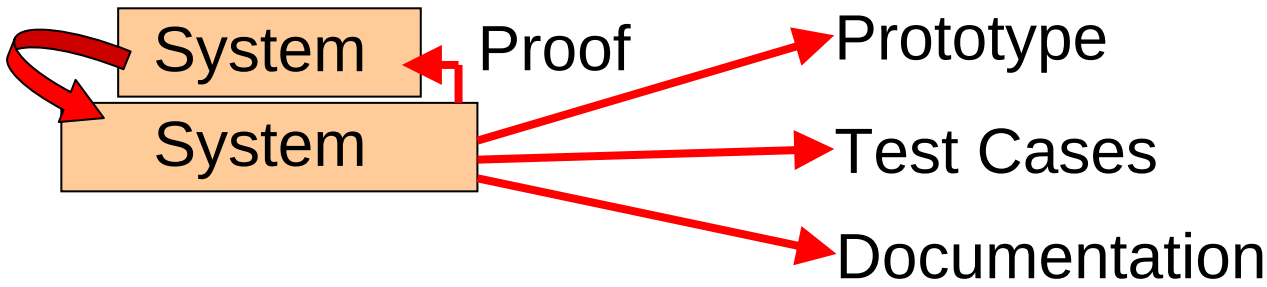
Model Driven Engineering (MDE)



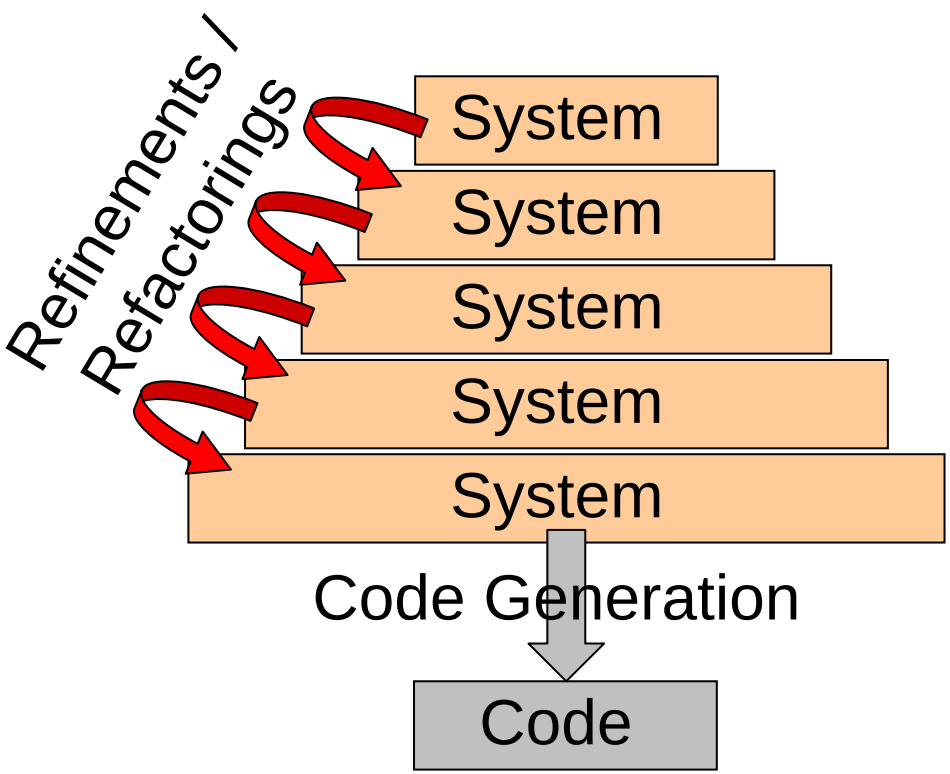
"From contemplative to productive"



MDE Step



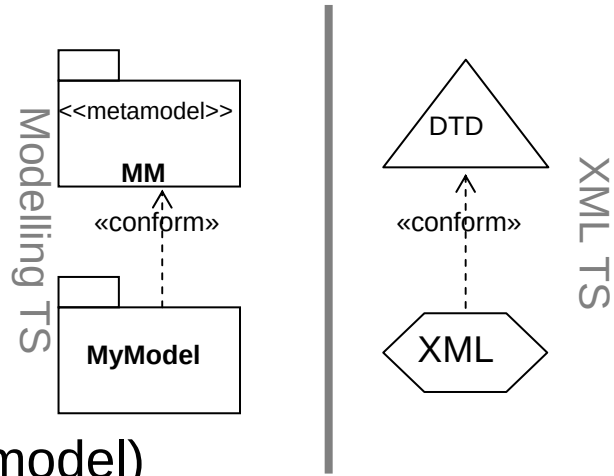
Questions Raised



- What language for describing systems ?
  - E.g. E-Banking
    - From Use Cases…
    - …to Oracle Schema + EJB + JSP
  - E.g. Subway
    - From Use Cases…
    - … to Wireless JavaCard
  - …
- One unique language is not enough !
  - Domain Specific Languages
  - Define YOURSELF the right level of abstraction !

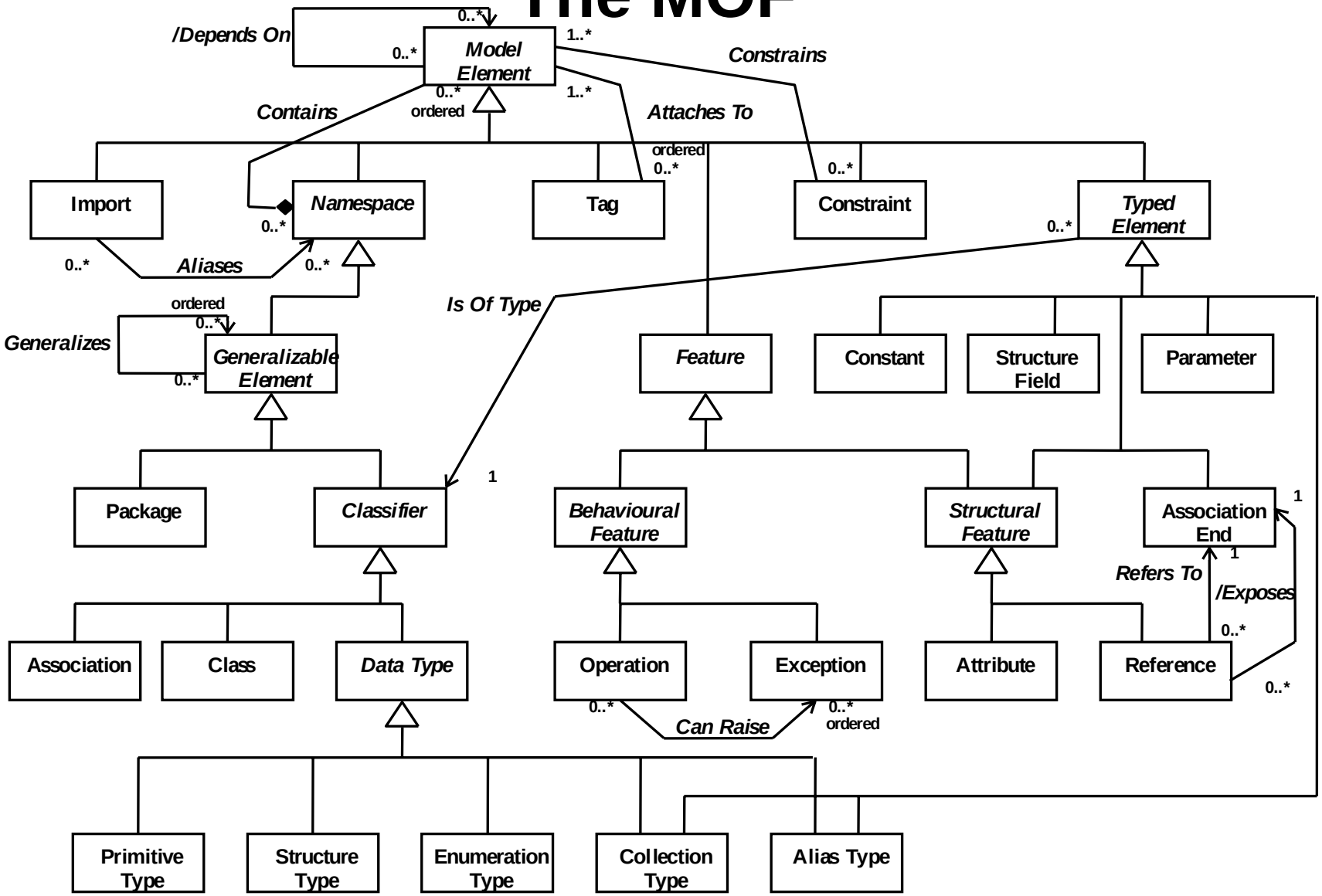
Contents

Metamodel

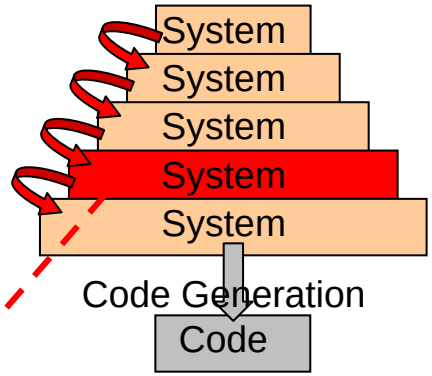


- Abstract Syntax for
  - Models
  - Domain Specific Languages
- Metamodelling Language
  - Needs an Abstract Syntax (Metametamodel)
  - A Metametamodel is a Language
    - Should be possible to describe it by itself !
- Remark: for a CASE tool, data are models
  - UML, Music Scores, Equipment of Industrial Systems, …
  - Communication among CASE tools with models
    - Requires knowledge of the surrounding metamodel
  - Metamodels structure model repositories as database schemas structure databases

The MOF


/Depends On
0..*
0..*
Model Element
1..*
Constrains
Contains
0..*
ordered
1..*
Attaches To
ordered
0..*
0..*
Import
Namespace
Tag
Constraint
Typed Element
0..*
0..*
Aliases
0..*
ordered
0..*
Generalizes
Generalizable Element
0..*
Is Of Type
Feature
Constant
Structure Field
Parameter
Package
Classifier
1
Behavioural Feature
Structural Feature
Association End
1
1
Refers To
/Exposes
0..*
Association
Class
Data Type
Operation
Exception
Attribute
Reference
0..*
0..*
Can Raise
0..*
ordered
Primitive Type
Structure Type
Enumeration Type
Collection Type
Alias Type

MOF : Layers



Let’s take a simple example of metamodel

«conforms»

«metamodel»
DataBaseMM

Element

NamedElement
name : string

TypedElement

typedElement
*
A_type_typedElement
0..1 type

DataBase

dataBase
table
0..1 *A_table_dataBase* *

Table

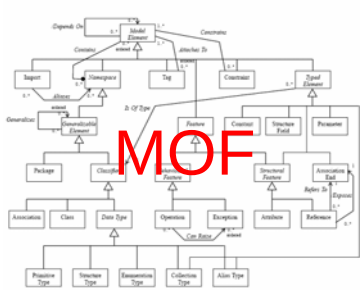
table
column
0..1 *A_column_table* *

Column
isPrimaryKey : boolean

DataType

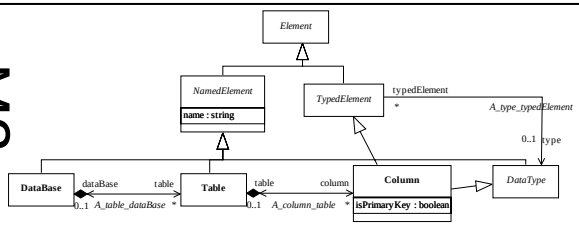
MOF : Layers

M3



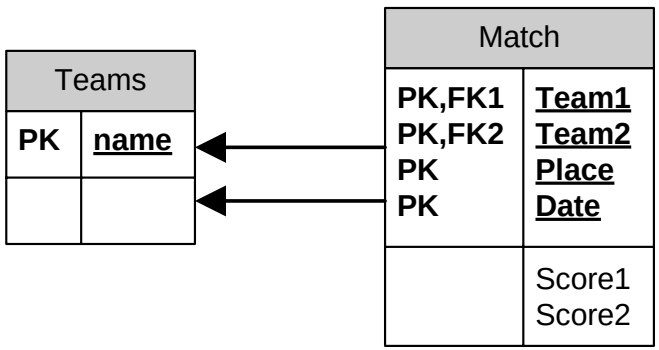
Metametamodel

M2



Metamodel

M1



Model

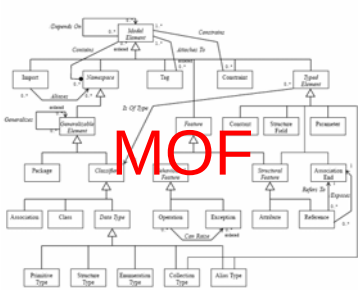
M0

Name
France
Switzerland
Germany

Real Life

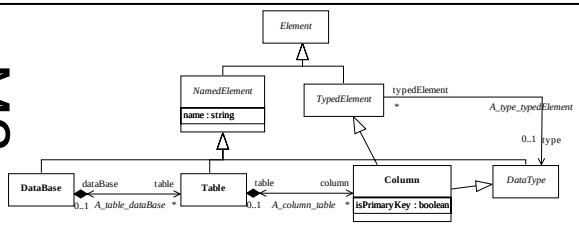
MOF : Layers

M3



Metametamodel

M2



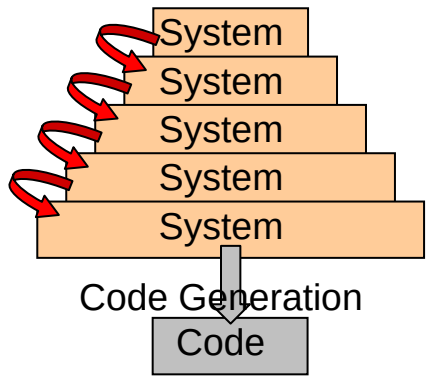
Metamodel

M1

Teams	
PK	**name**

Match	
PK,FK1 **PK,FK2** **PK** **PK**	**Team1** **Team2** **Place** **Date**
	Score1 Score2

Model



M0

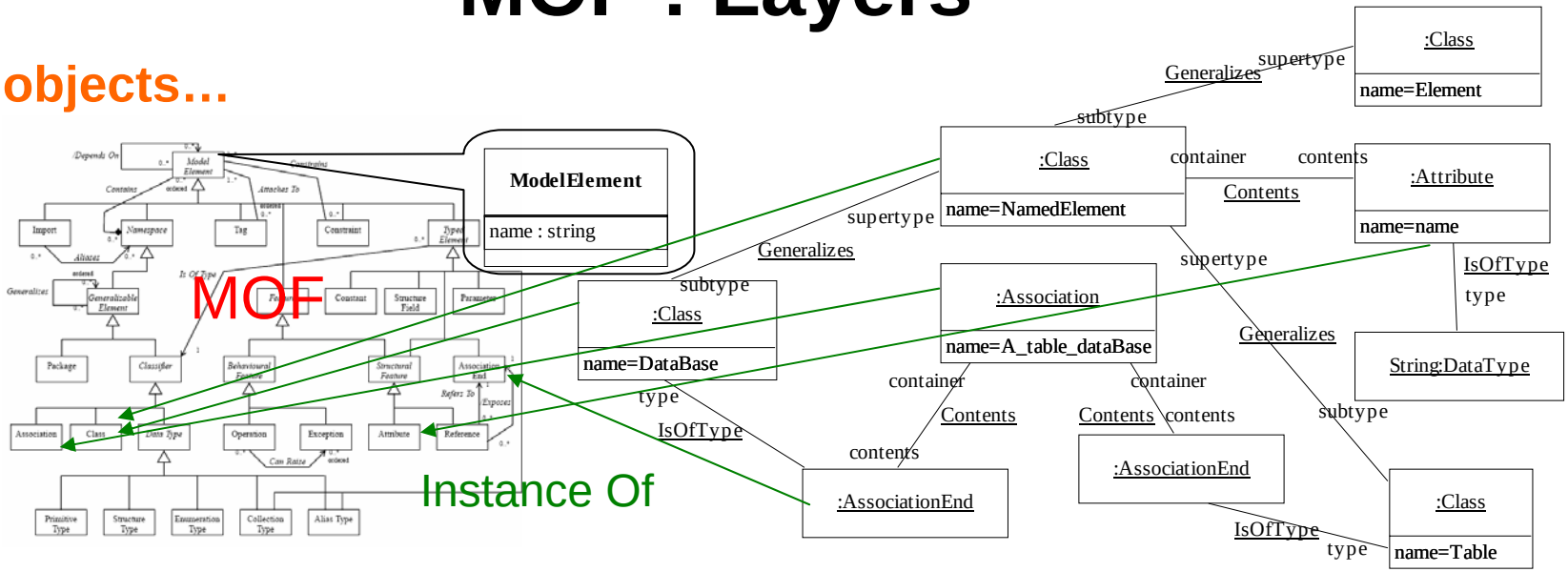
Name
France
Switzerland
Germany

"Real" Life

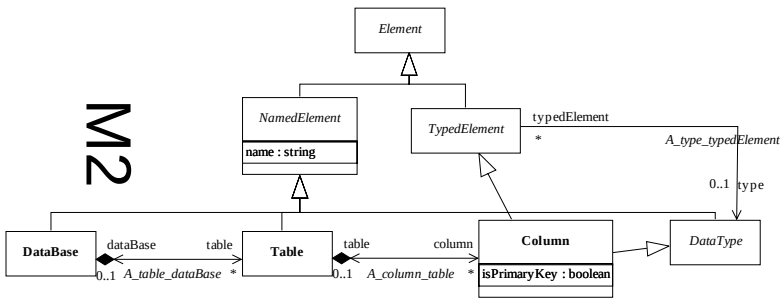
MOF : Layers

M3 objects…

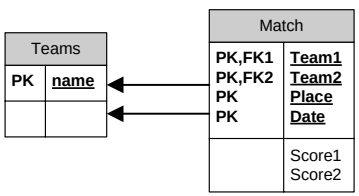
M3



M2

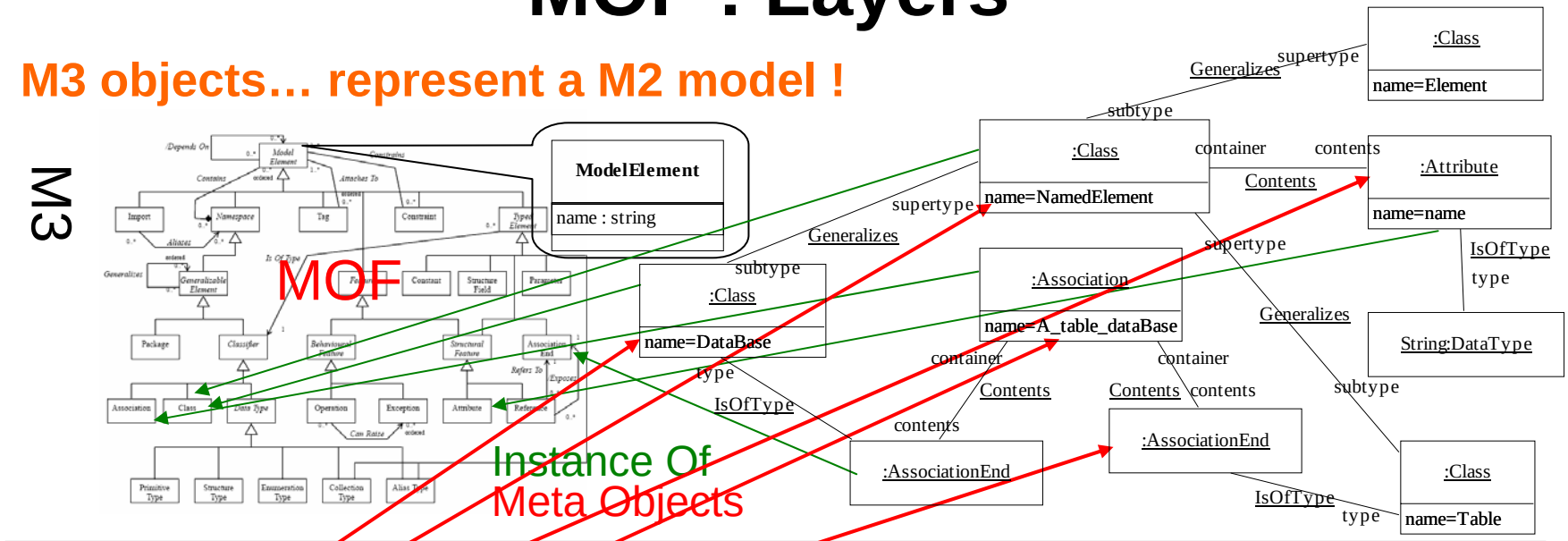


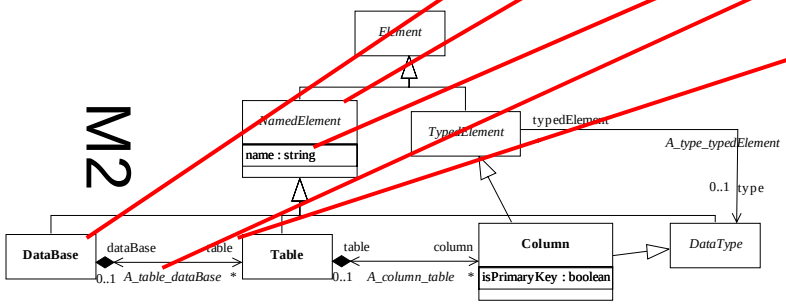
M1

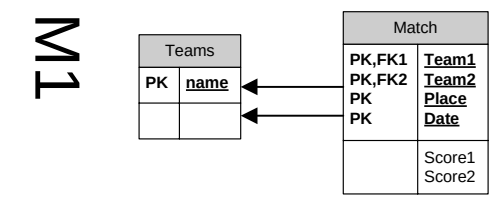


MOF : Layers

M3 objects… represent a M2 model !

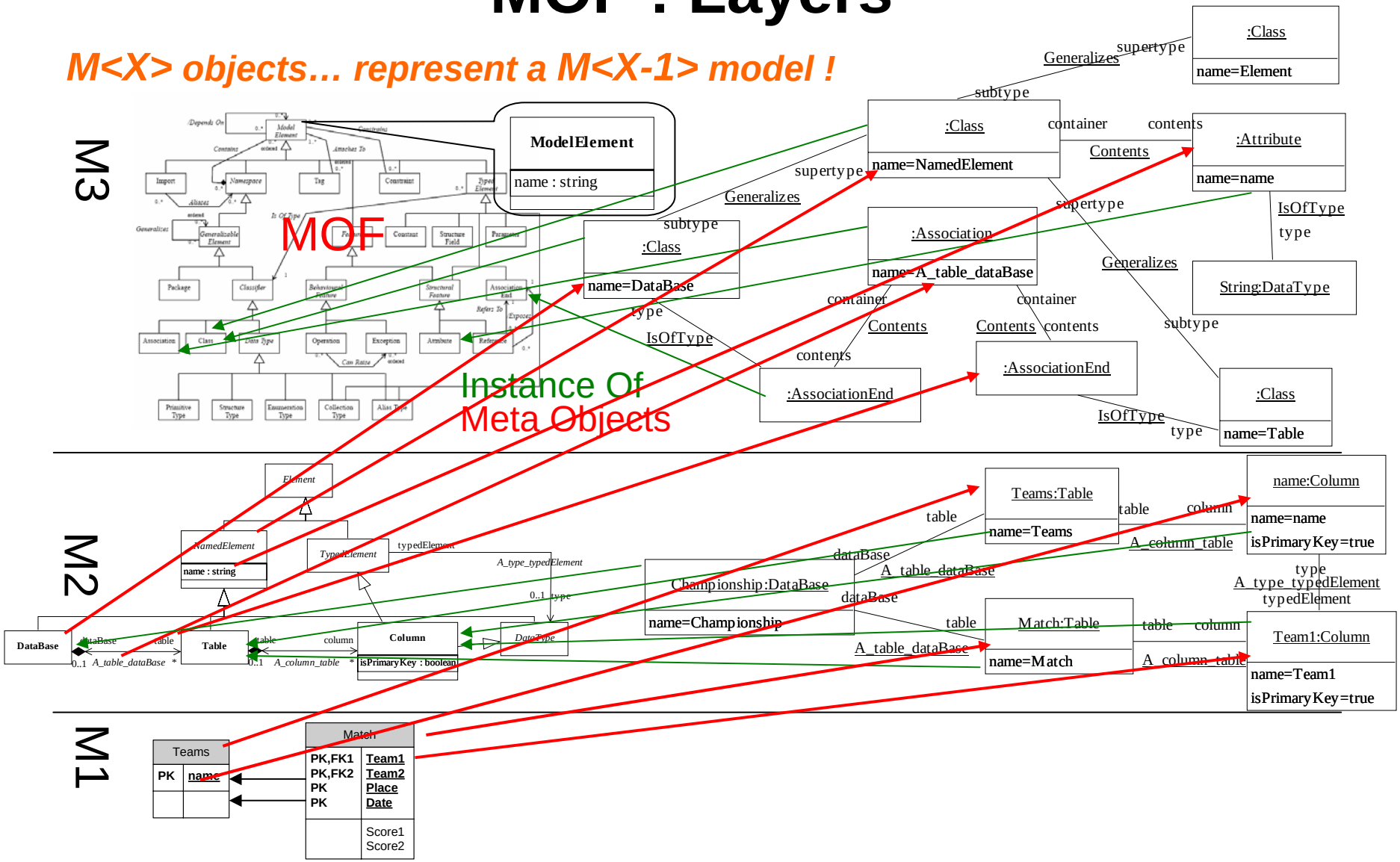






MOF : Layers

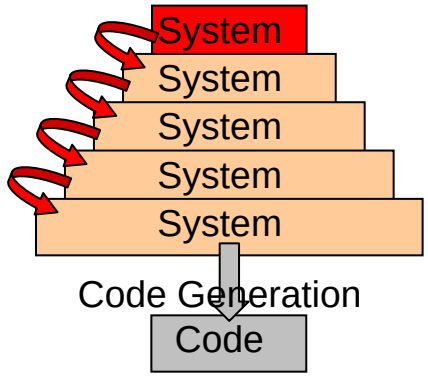
M<X> objects… represent a M<X-1> model !



Contents

- Software engineering
- Modelling
- Metamodelling
- A case study
- Concluding remarks

Step 1 : Use-Cases

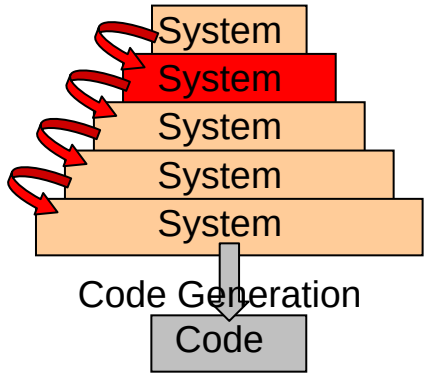


- Modelling a Genealogy website.

- 1st: Requirement elicitation
  - I want a web application that shows me the family of somebody
    - Wife
    - Husband
    - Daughters
    - Sons
  - The Web application must let me enter family information

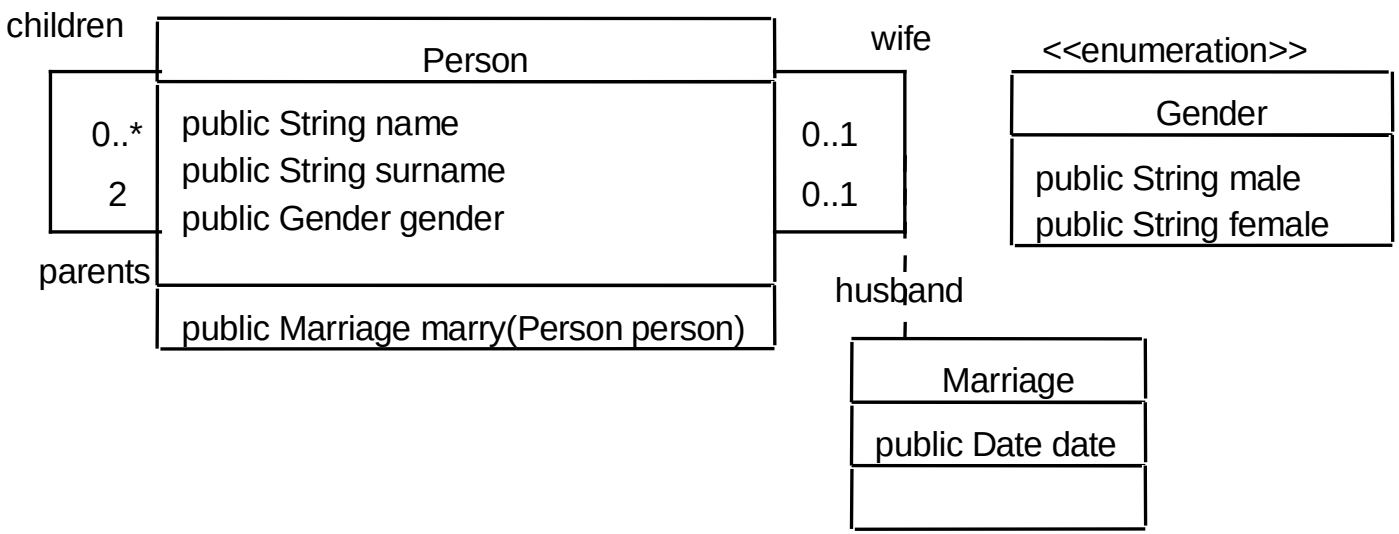
  - Could be captured more precisely in Use Cases and Sequence Diagrams…

Step 2 : Domain

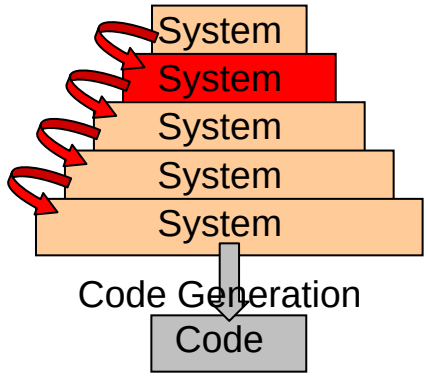


- Modelling a Genealogy website.

- 2nd: Business Model
  - A class diagram
    - Metamodel : UML core

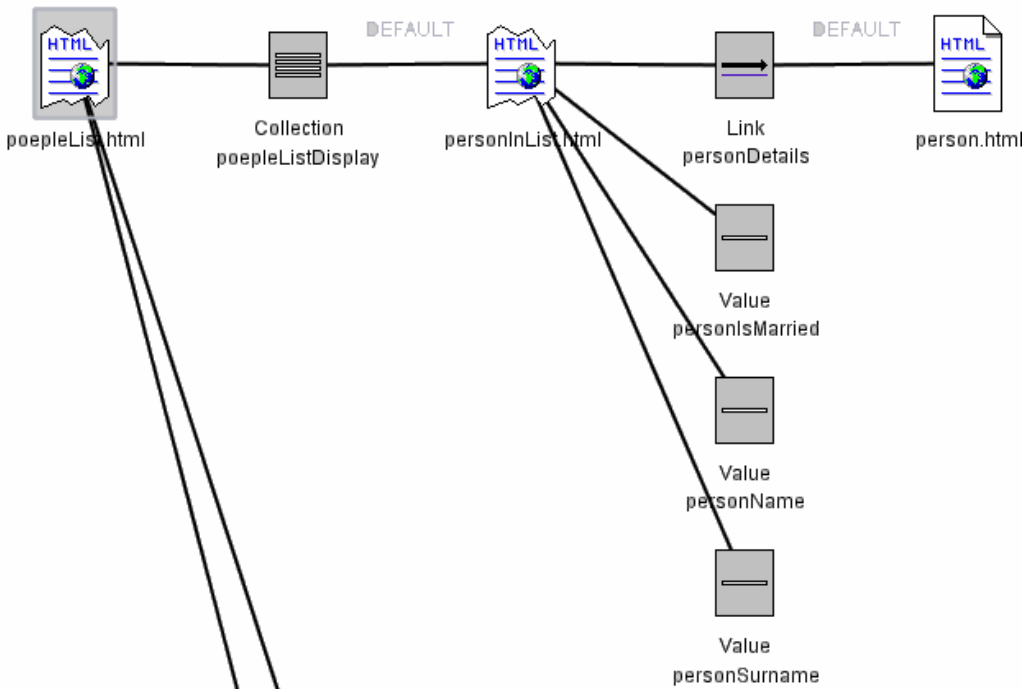


Step 3 : Cinematic

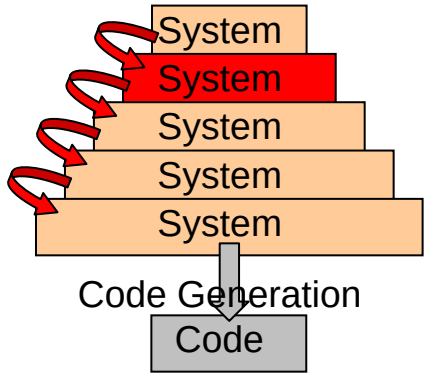


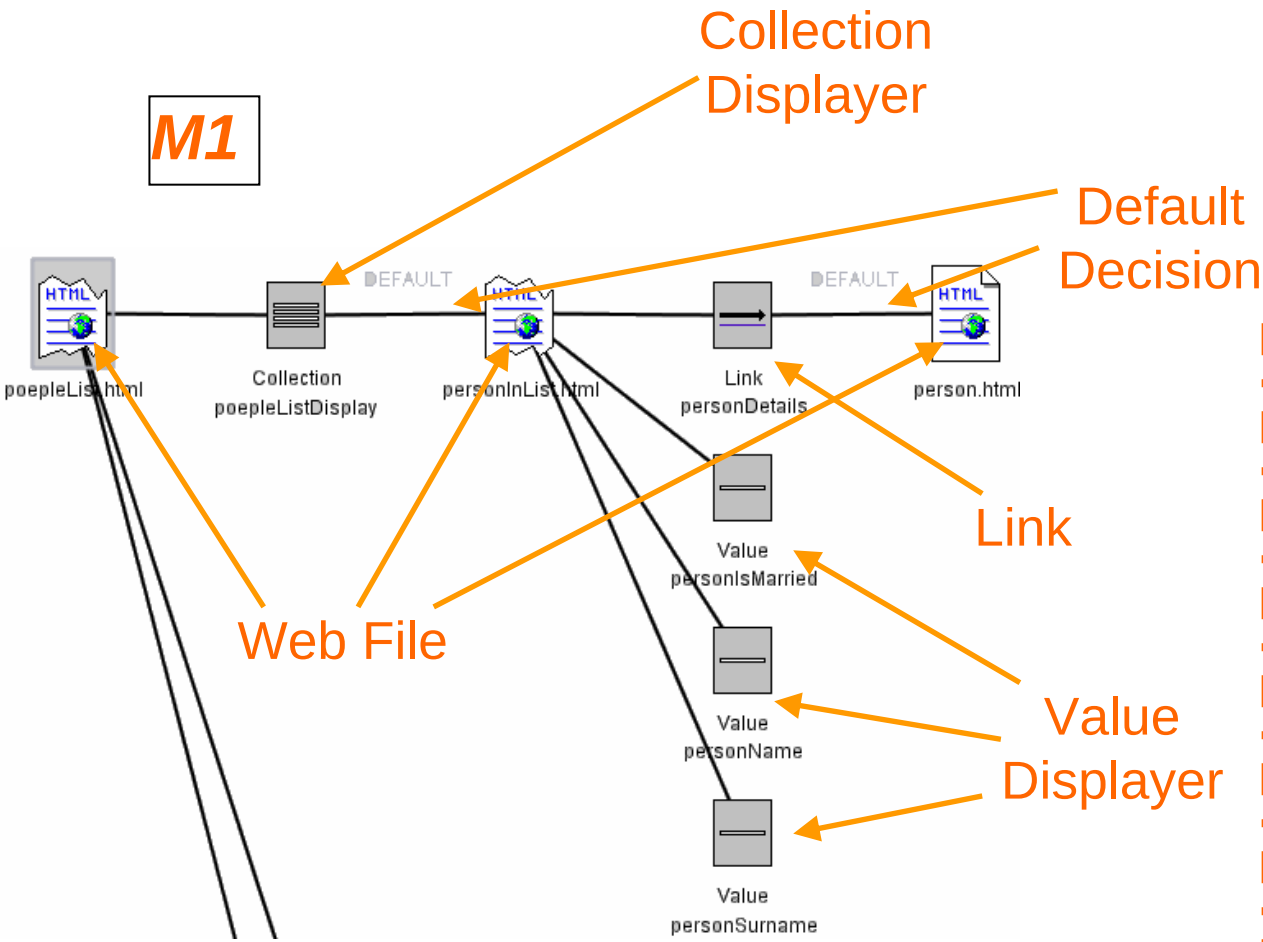
- Modelling a Genealogy website.

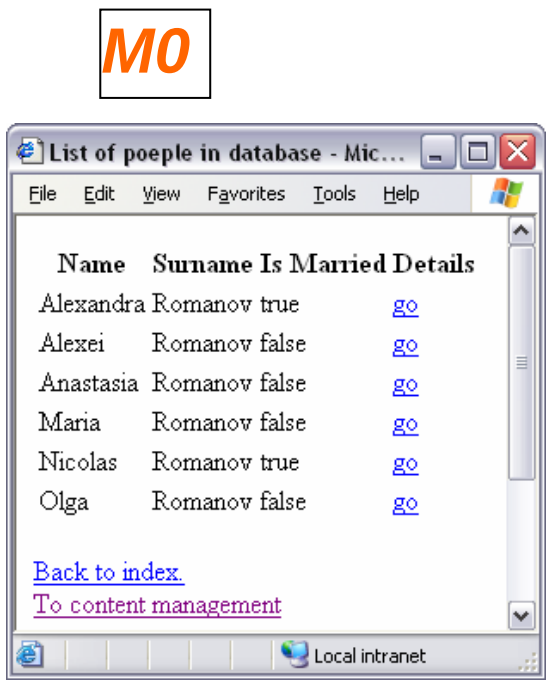
- 3rd: Web site cinematic…
  - A DSL : Netsilon Hypertext



What is that new Model ?

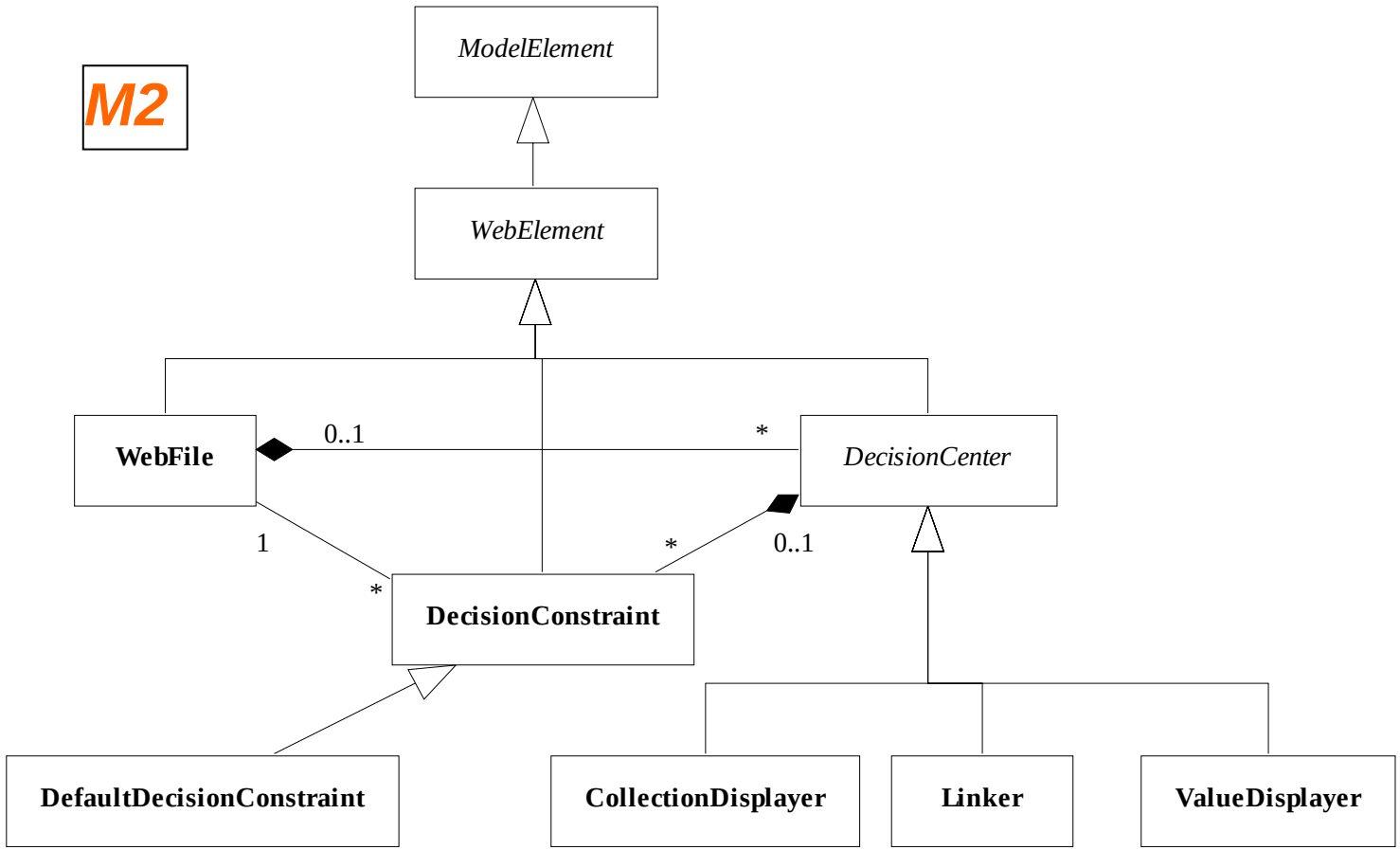




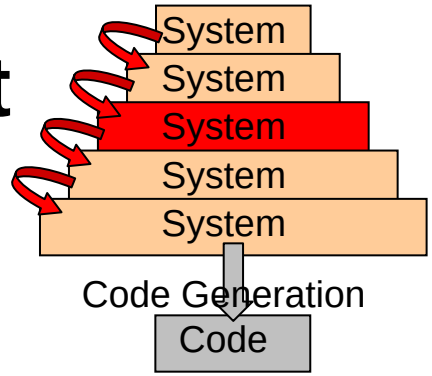


Cinematic : Metamodel

- Netsilon Hypertext Metamodel

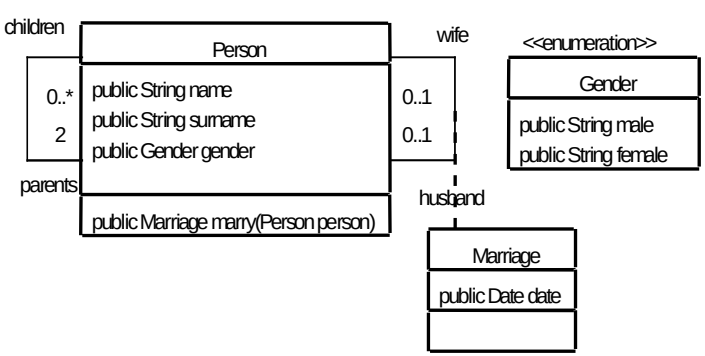


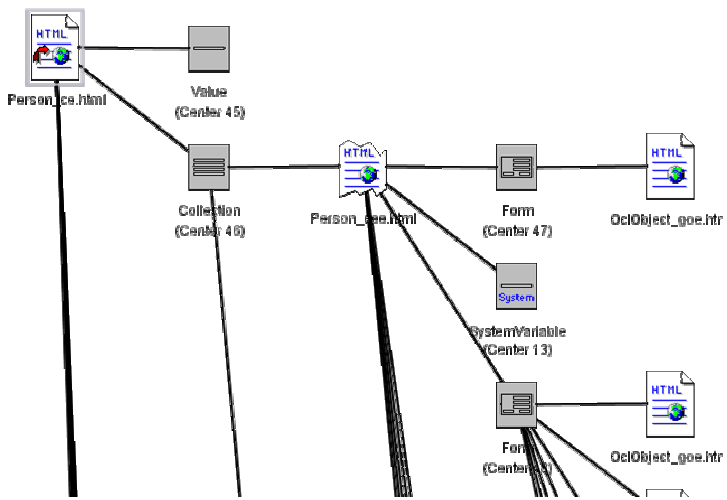
Step 4: Data Management



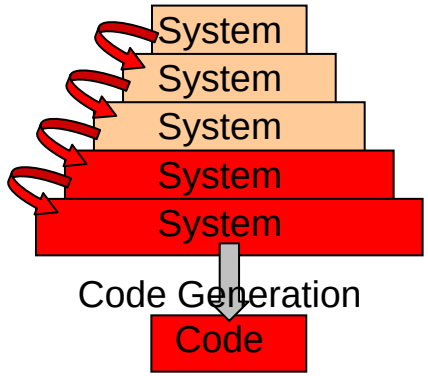
- Modelling a Genealogy website.

- 4th: Generate data management part of the web application
  - Automatic model transformation from the business model to web application hypertext model
    - Class Diagram -> Hypertext Model





Lasts steps : Platforms

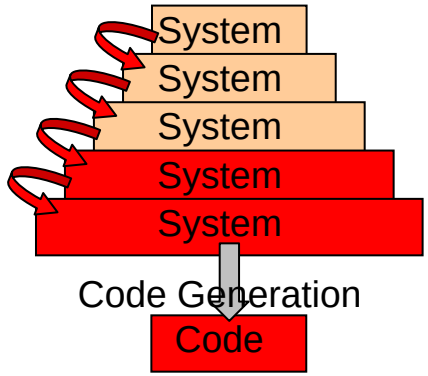


- Modelling a Genealogy website.

- 5th: Choose platforms…
  - Now and only now !
  - *JSP / MySQL*
  - Generate code
  - Not much to develop here but automatable optimizations…

- 6th: Deploy !
  - On server www.genealogy.com/family
  - On database server.genealogy.com named *Family*

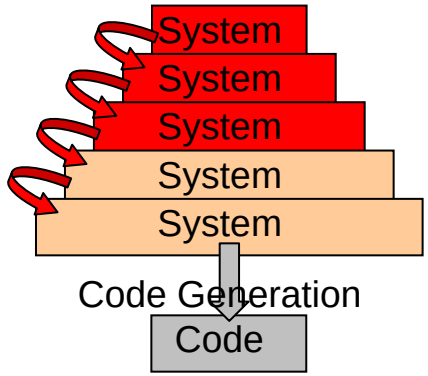
After : Maintenance



- Modelling a Genealogy website.

- 7th A: Maintain !
  - The application is victim of its success
  - Platform have to be changed !
  - *Servlet / Oracle*
  - Repeat phases
    - 5 : “Choose platform and generate code”
    - 6 : “Deploy”
    - (needs data migration)

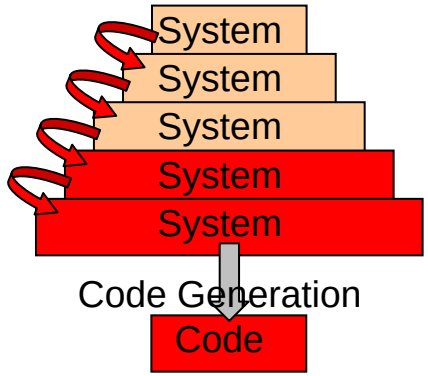
After : Maintenance



- Modelling a Genealogy website.

- 7th B: Maintain !
  - A new requirement appeared !
  - *Visibility restricted to one's own family*
  - Redesign appropriate phases
    - Guided by the upper layer
  - Regenerate / Redeploy automatically

After : Maintenance



- Modelling a Genealogy website.

- 7th C: Maintain !
  - Your system must interoperate with other systems !
  - *Remote calls from CORBA*
  - Select a new platform *(CORBA)*
  - Regenerate / Redeploy automatically

Contents

- Software engineering
- Modelling
- Metamodelling
- A case study
- Concluding remarks
  - Who is involved
  - What is Metamodelling
  - What is NOT Metamodelling

Who is Involved ?

- Actors
  - Domain specialists
  - SE Method developers
  - NOT system developers
- Developed once, often used
  - There are much more programs than languages !

What is Metamodelling

- Make models machine friendly beside human friendly
  - Enables tool interoperability at model level
  - A first step for automating (at least partially) incremental and reactive refinements

- Define an ontology for a class of models
  - Vocabulary (terms)
  - Taxonomy (relations between terms)
  - Notation (model depicting)
  - Epistemology (what is represented) + Semantics
  - Can define extension mechanisms (profiles)

Needs More Maturity

- Make models machine friendly beside human friendly
  - Enables tool ***interoperability*** at model level
  - A first step for automating (at least partially) ***incremental and reactive refinements***

- Define an ontology for a class of models
  - Vocabulary (terms)
  - Taxonomy (relations between terms)
  - ***Notation*** (model depicting)
  - ***Epistemology*** (what is represented) + ***Semantics***
  - Can define extension mechanisms (profiles)

Unfortunately, standards are not going that far… at the moment !

What is Not Metamodelling

- Defining relationship among models
  - Model Transformation
    - Bidirectional
    - Traceable
    - Defined on metamodels
  - Code
    - Generation
    - Reverse
    - May be linked to Concrete Syntax definition
      - E.g. there are metamodels for Java

Thank you for your attention !

- Any question ?

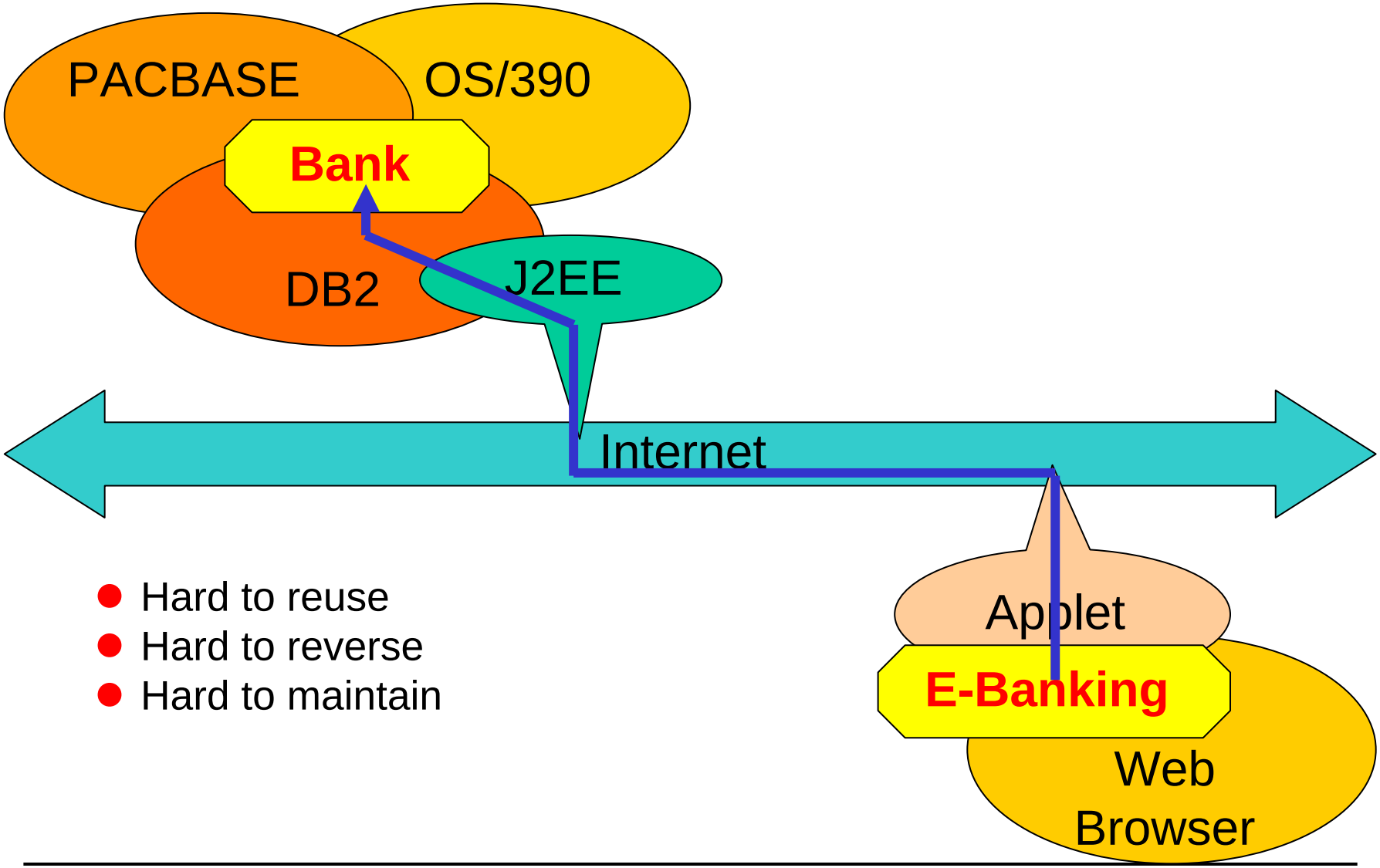
Annexes

- Variability
- A Typical Variability Example
- The Fondue Toolset Architecture
- MOF Model Interchange
- Ontology Overview

Software Lifecycle and Variability

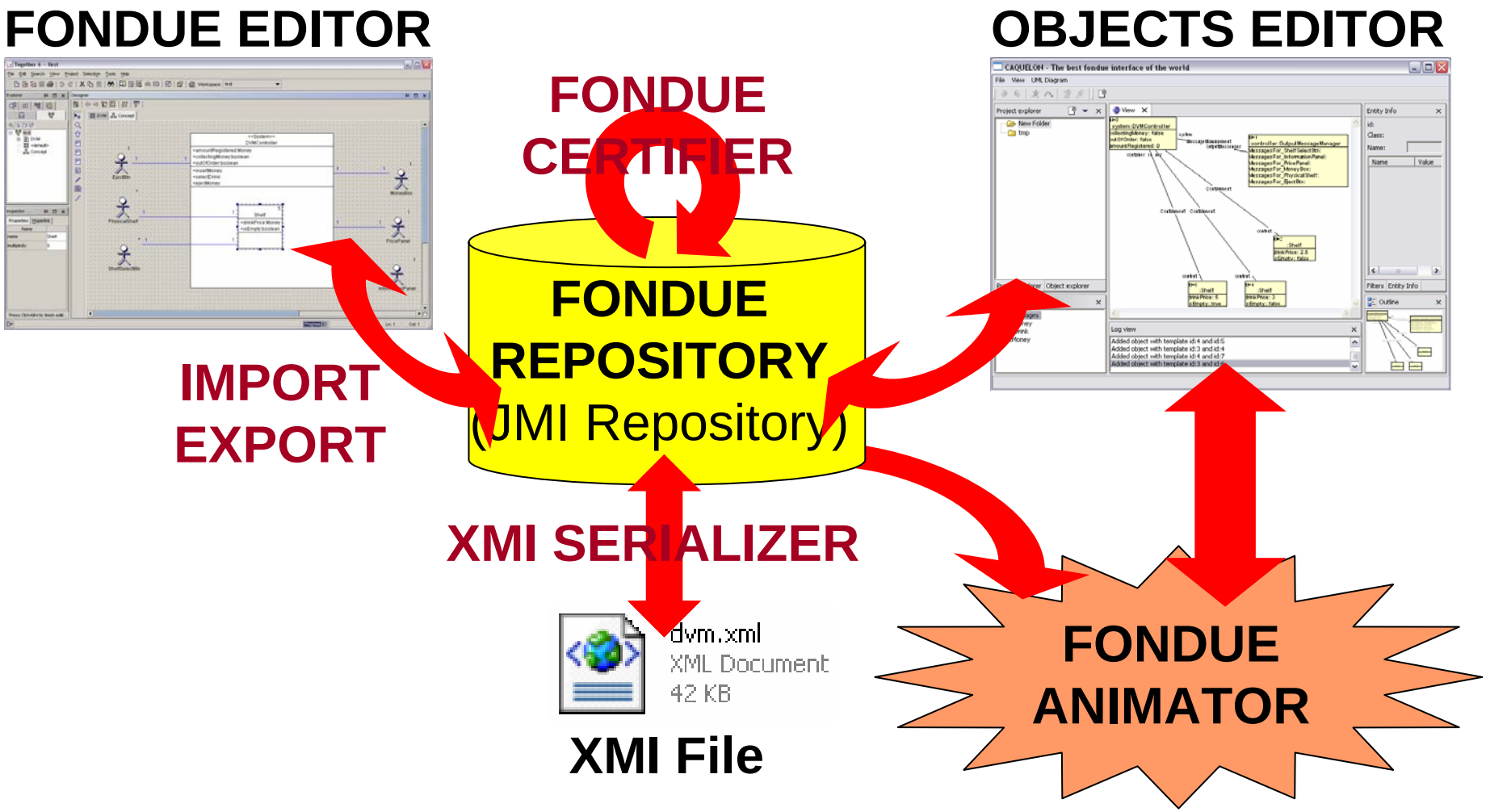
●Requirement elicitation	Nokia: ●50% changes after “frozen” ●60% of these changes at least twice
●Analysis / Design	
●Implementation	●Compilers ●Frameworks
●Verification / Test	
●Deployment	●Middleware ●Databases ●Application server ●Hardware
●Maintenance	●During years (e.g. bank and insurance companies) ●Turnover ●Previous variabilities

Typical Example



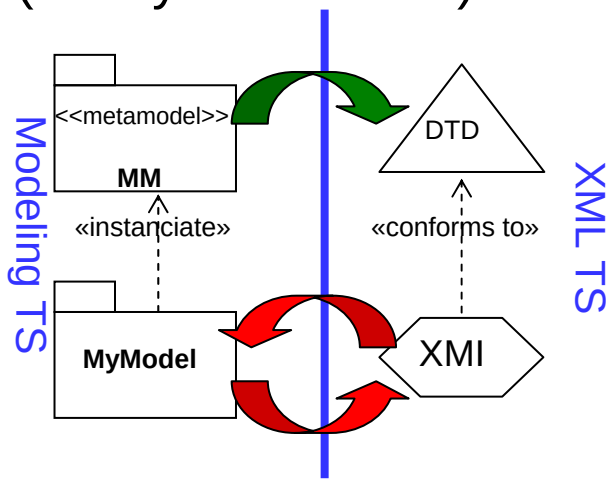
- Hard to reuse
- Hard to reverse
- Hard to maintain

Fondue Builder Architecture Overview



MOF : Model interchange (XMI)

- Model interchange is standardized
- Should take into account models of any kinds (of any metamodel)
- XMI is XML => it needs a schema
- Schema is given by the M(X+1) level !
  - Tools generate a DTD from a metamodel
  - Tools load / store models from / to XMI
  - An “XMI model” is valid for a given metamodel
    - XMI is a language template



- Remark
  - 3 versions of XMI (1.0, 1.1, 1.2)
  - Many versions of metamodel (UML : 0.9, 1.0, 1.3, 1.4, 2.0…)
  - Tools interpret the XMI standard as they wish !
  - XMI possibilities for a same model (of a given metamodel) : Cartesian product of
    - XMI version
    - Metamodel version
    - Tool

Ontology Overview

- Describes
  - *Vocabulary*: set of terms
  - *Taxonomy*: relationships between terms (parent / child, …)
  - Thesaurus: equivalences between terms
- Usages – *Jasper R. and Uschold M.*
  - *Neural authoring*: knowledge of an expert system
  - *Specification*: core of a tool (Poseidon, Fondue Toolset, …)
  - *Common access to information*: interoperability (tools, DBs, …)
  - *Ontology-Based Search*: data mining (Semantic Web, …)
- Layers
  - L0: Objects described by an ontology
  - L1: The ontology itself
  - L2: The language for describing ontology (OWL, IDL, KAON, …)

Ontology Overview

M3		L2	L2-L1	L2-L1-L0
M2	L2	L1	L0	**The Metametamodelling ontology**
M1	L1	L0	**The Metamodelling ontology**	
M0	L0	**Modelling (metamodelling)**		

Data Management (semantic web, warehousing,…)