ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Component-Based Language Engineering

Frédéric Fondement

Swiss Federal Institute of Technology, Lausanne
Software Engineering Laboratory

SMV & LGL Seminar
Les Diablerets, June 6th, 2005

Contents

- Language Engineering
- Language Components
- Identified Kinds of Components
- Tuning
- Outlook

Contents

- Language Engineering
- Language Components
- Identified Kinds of Components
- Tuning
- Outlook

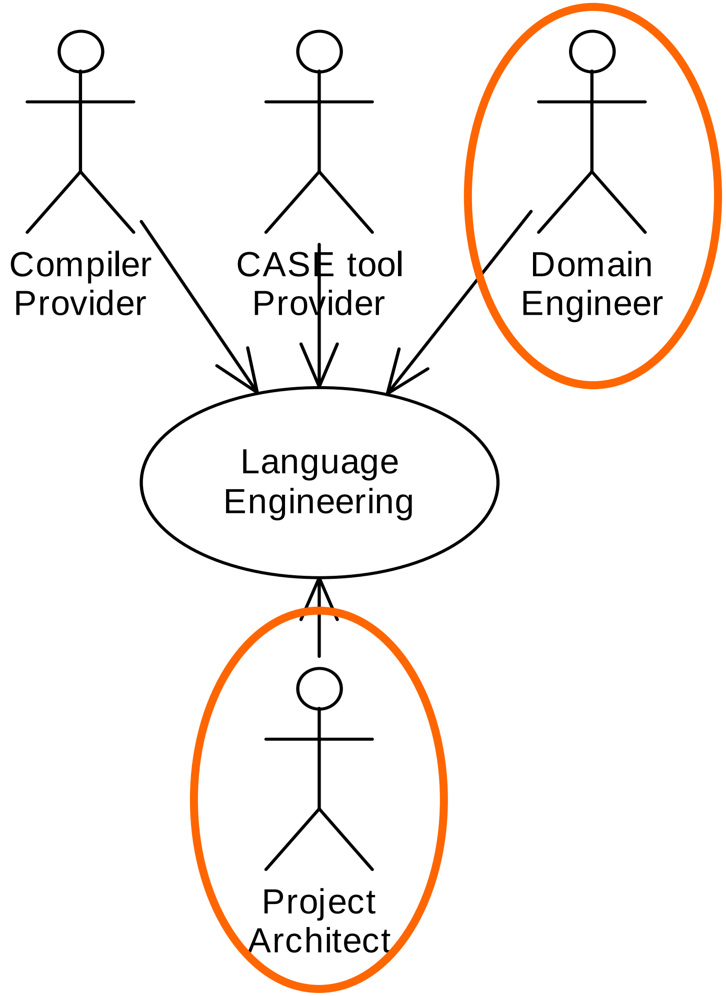
Language Engineering

- Very Heart of Model/Language Driven Engineering

- Well-Known **Domain Abstractions** (Ladder / Phototool / Architecture…)

  vs.

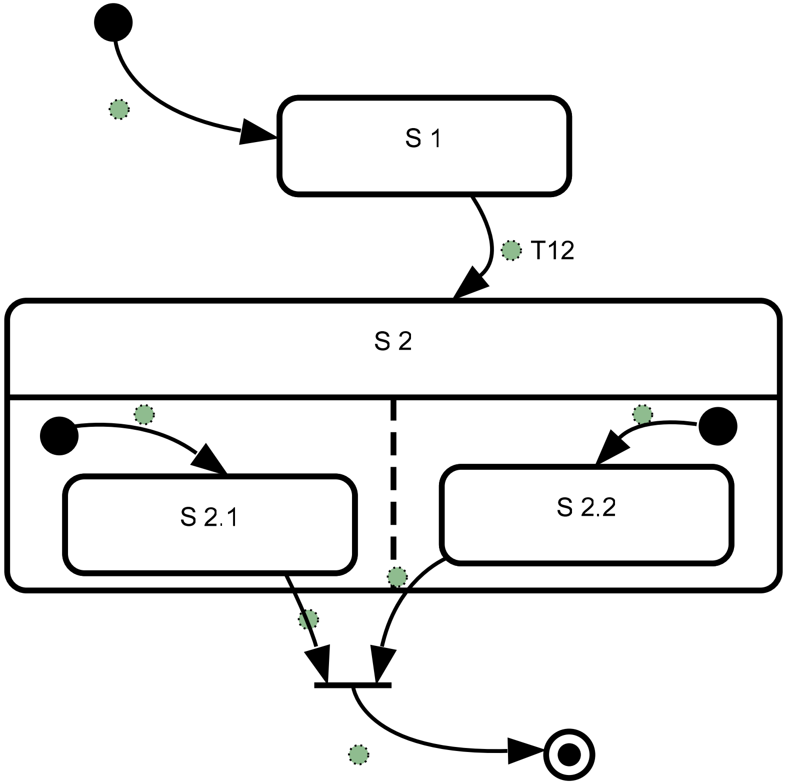
  (more or less) adapted **Generic Purpose Languages** (UML/…)



- Proliferation of Languages

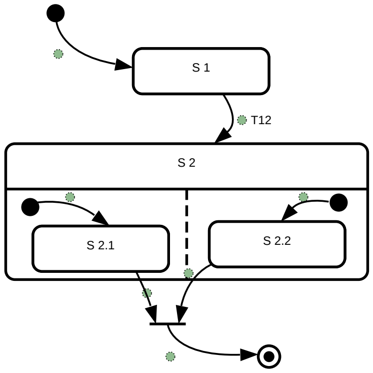
Language Construction

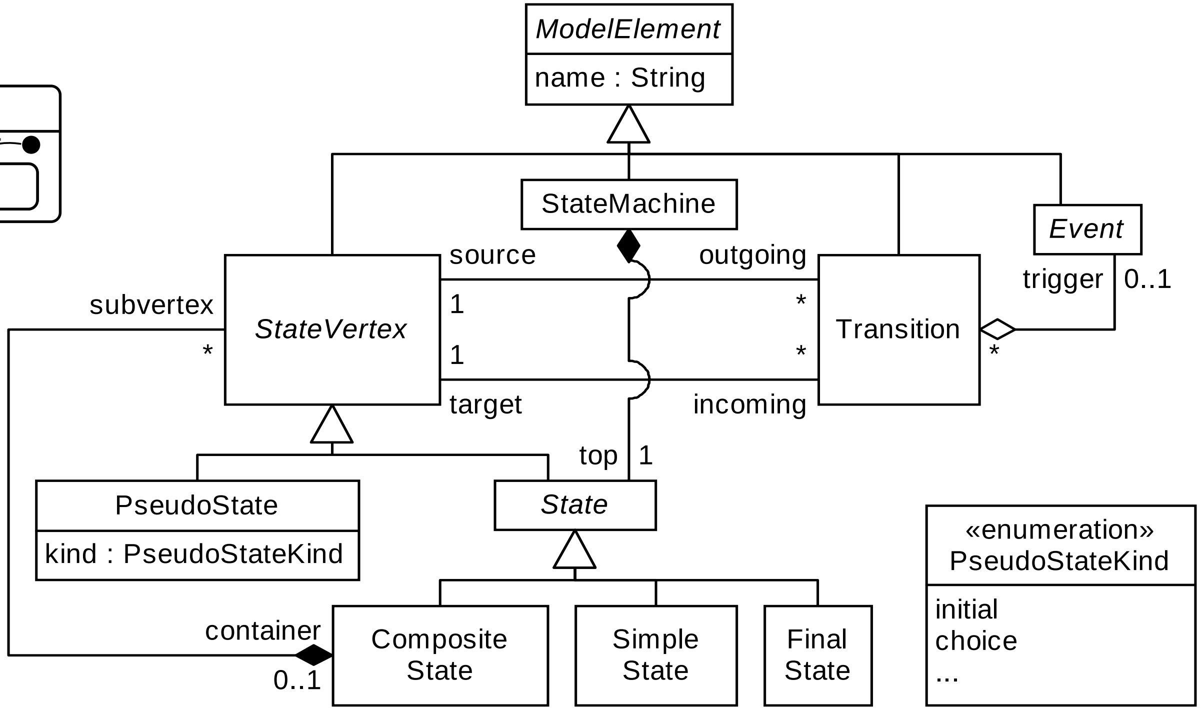
Concrete Syntax



Language Construction

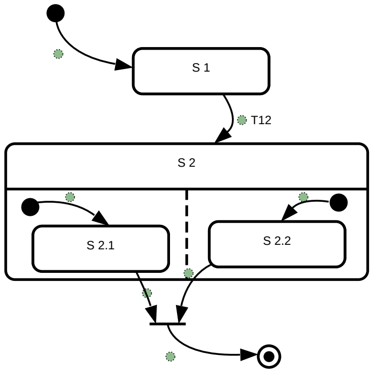
Concrete Syntax + Abstract Syntax

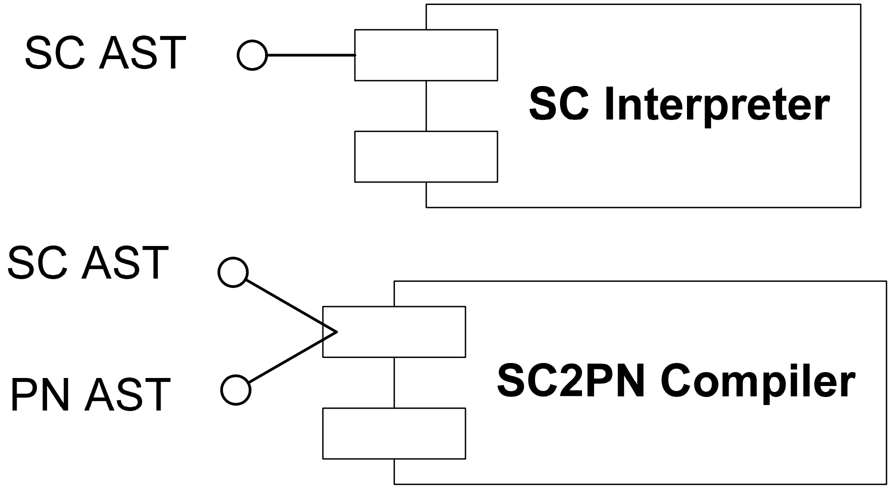




Language Construction

Concrete Syntax + Abstract Syntax + Semantics



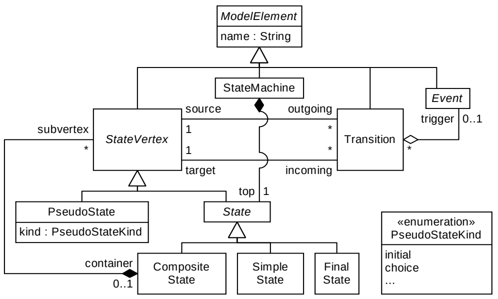

SC AST

SC Interpreter

SC AST

PN AST

SC2PN Compiler



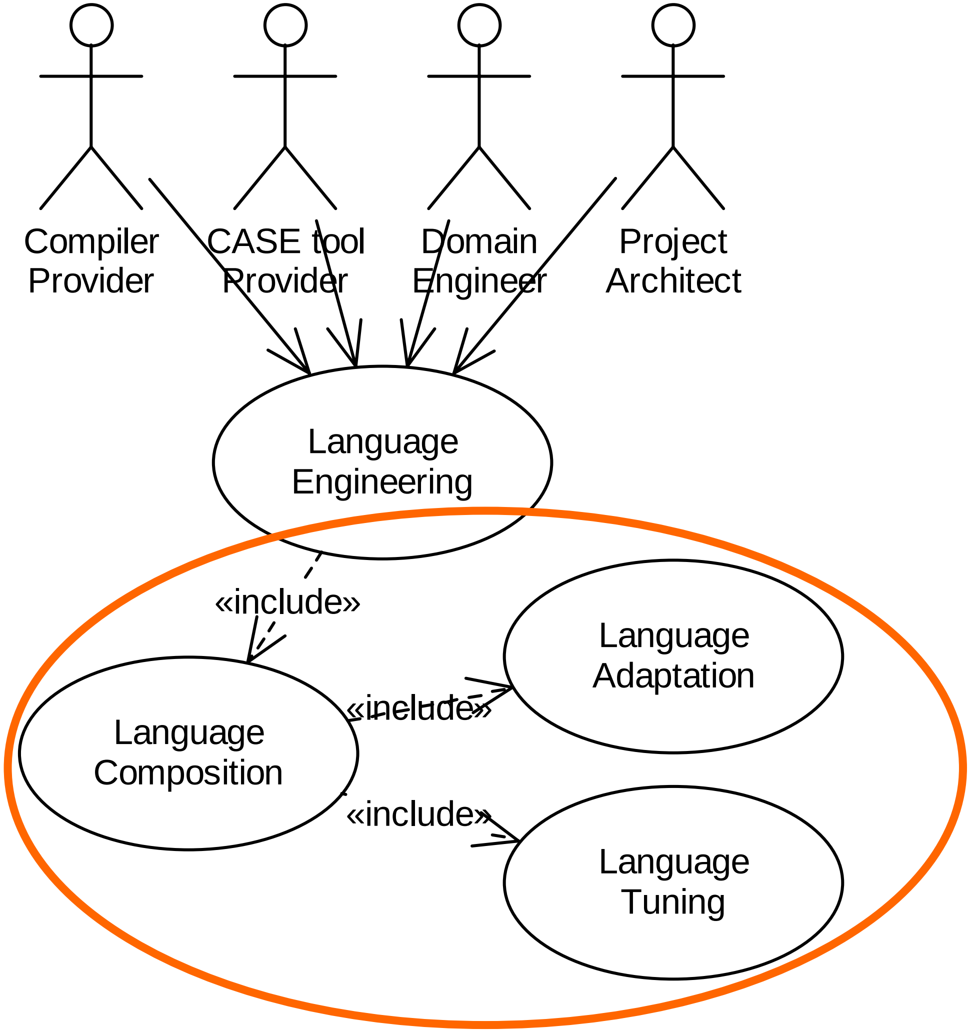
Problems in Language Engineering

- An Endeavour to Capture Knowledge in a Language
  - Concrete Syntax + Abstract Syntax + Semantics
- Built from Scratch
- Built from A to Z
- Recurrent Comparable Solutions
  - At syntax level
    - E.g. UML & MOF (incl. different versions)
  - At concepts levels
    - E.g. Merise & UML; Scenarios & MSC
  - At semantics level
    - E.g. OCL & SQL ; Java and Smalltalk objects
  - Evolution of Standards (versions)

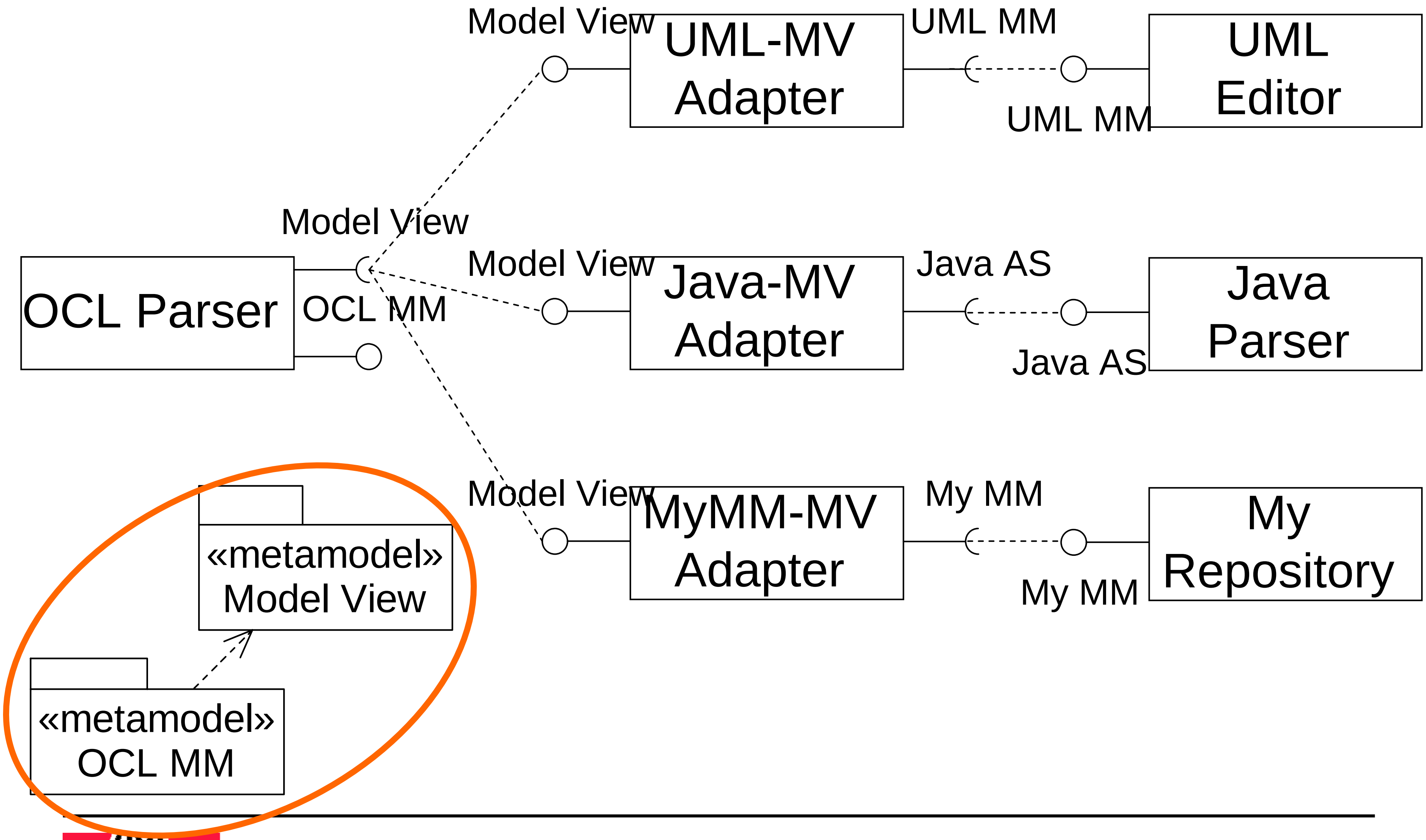
Contents

- Language Engineering
- Language Components
- Identified Kinds of Components
- Tuning
- Outlook

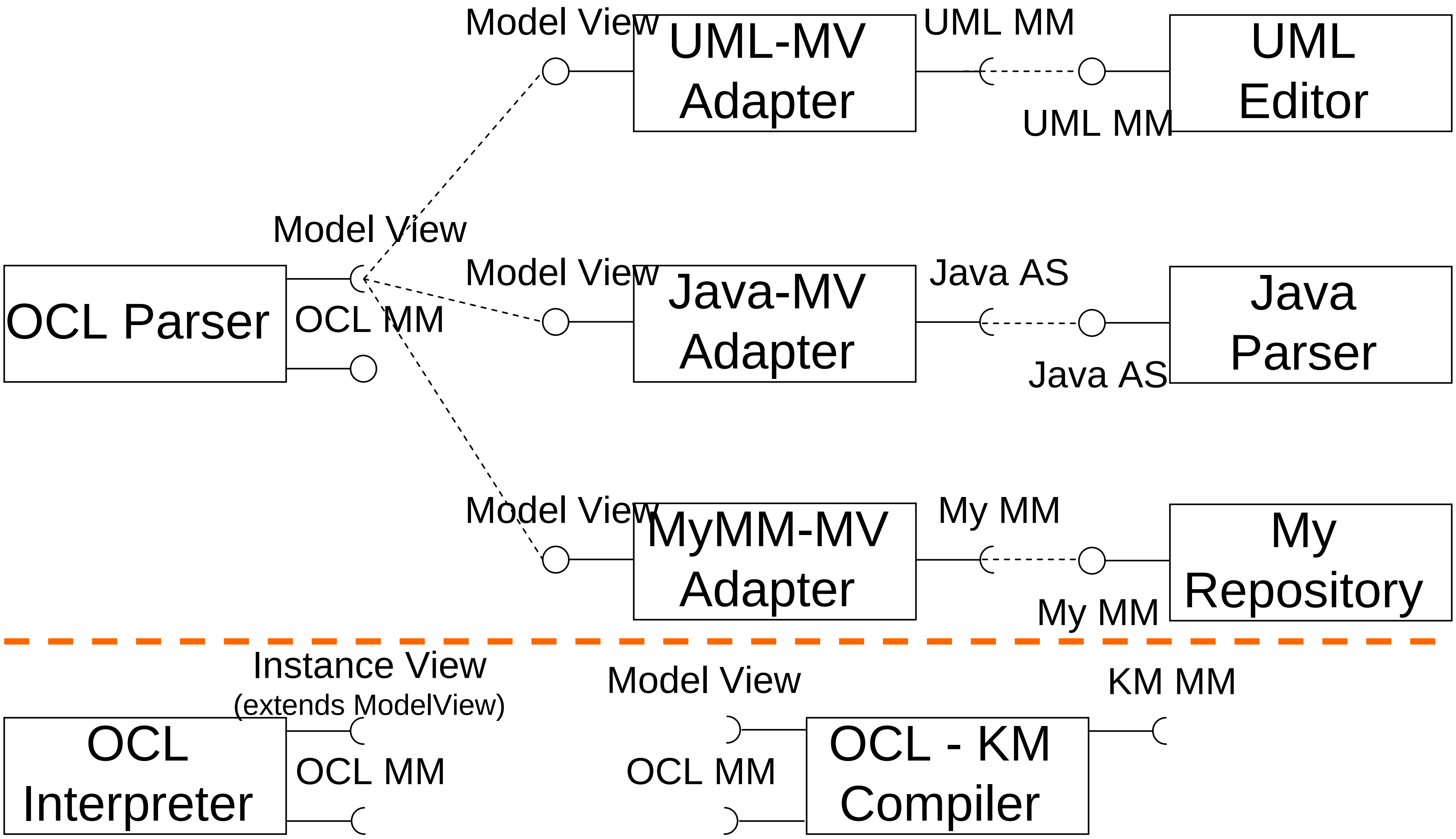
The Idea


Compiler Provider
CASE tool Provider
Domain Engineer
Project Architect
Language Engineering
«include»
Language Composition
«include»
Language Adaptation
«include»
Language Tuning

An Example : Generic OCL Framework



An Example : Generic OCL Framework

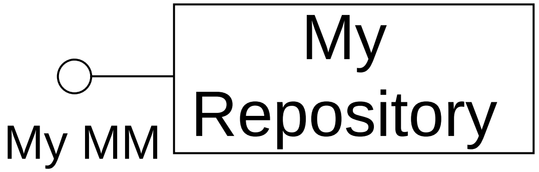


Contents

- Language Engineering
- Language Components
- Identified Kinds of Components
- Tuning
- Outlook

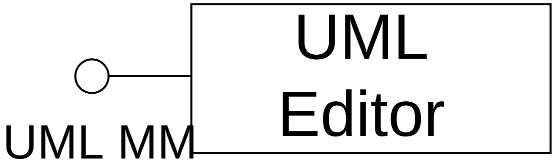
Kind Of Component : Repository

- Metamodel-Driven
- Already at work
  - XMI
  - JMI
  - …



Kind Of Component : Editors

- Environment to Edit / Render a Graphical Model
- (can integrate a repository)



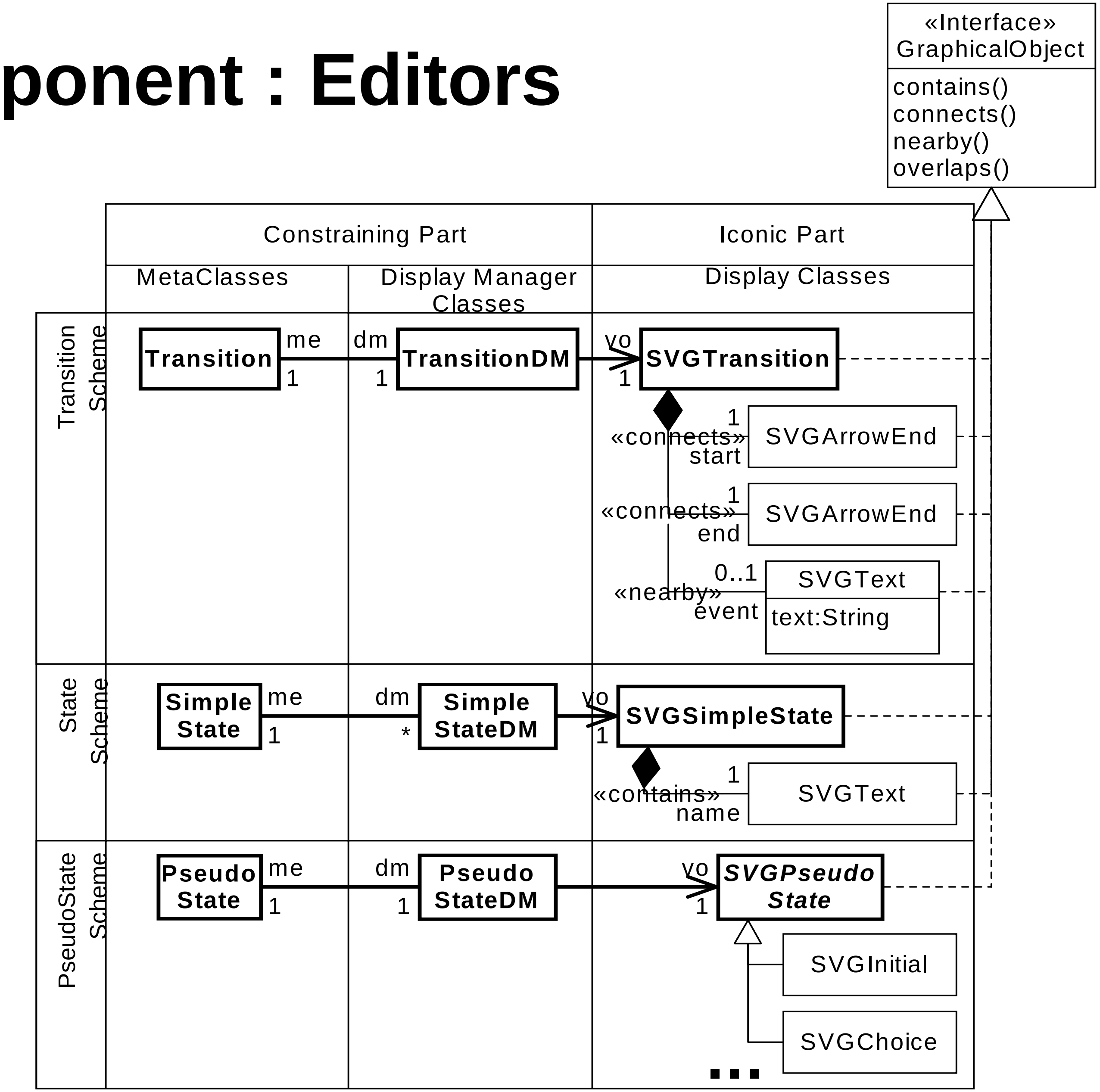
- Technologies
  - Specific CASE tools – hand-coded
  - Meta-Editors – Platform-specific

- Concrete syntax cannot be formalized !

Kind Of Component : Editors

Proposal to define (graphical) Concrete Syntaxes

- SVG templates
  - Properties
  - Behaviours
- Mapping model
- Consistency Constraints
  - MM vs. Representation
  - Spatial Relationships

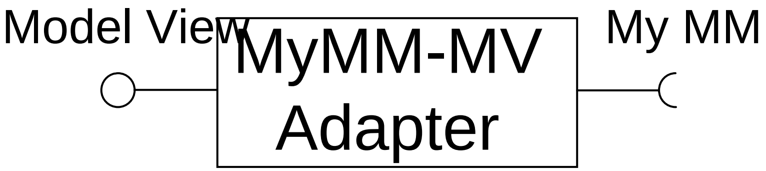


Kind Of Component : Adapters

- Isolation Layer between Components at Model Level
  - Reusability of components (incl. Model Transformations)
  - Implementation of Interfaces

- Maps Required Interfaces (Model View) with Actual Interfaces (UML / Java / MyMM / …)

- Could be realized by Model Transformation
  - Synchronization issues



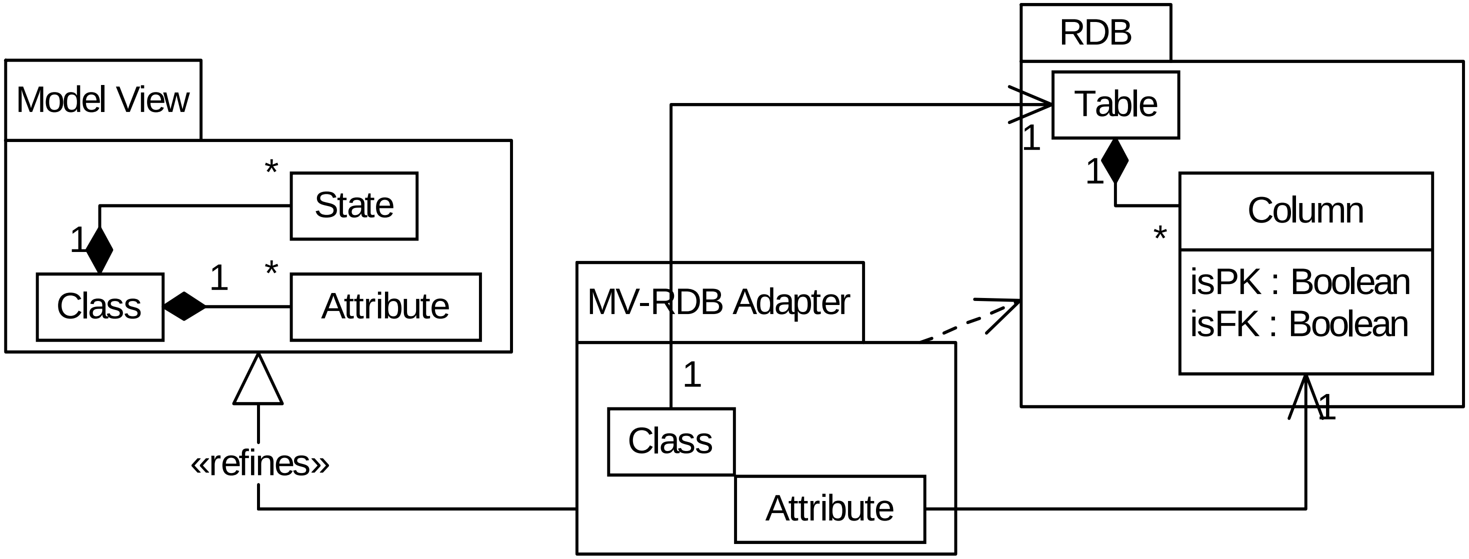
Model View

MyMM-MV Adapter

My MM

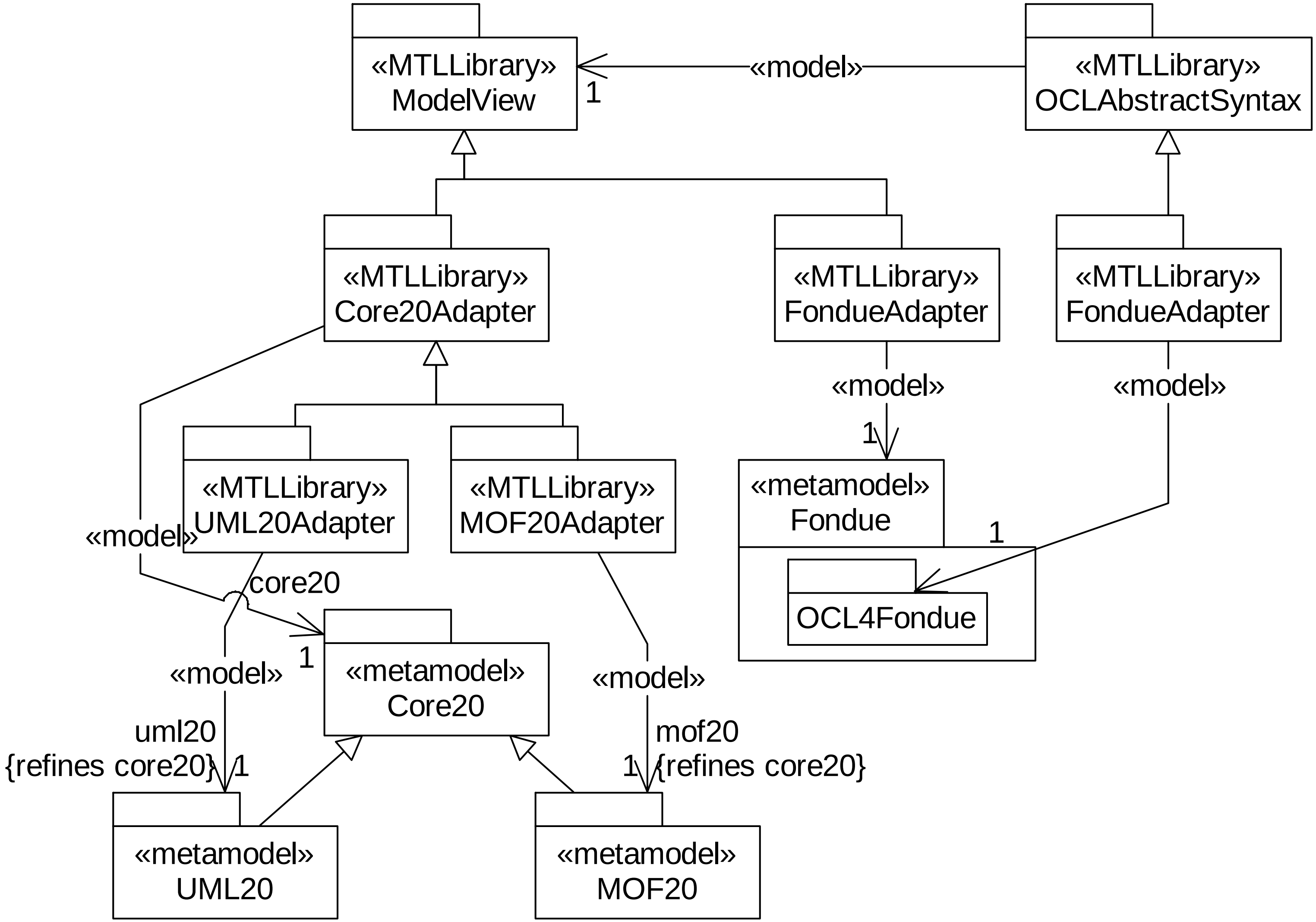
- Proposal : Use Refined Views
  - The View Pattern – at model level
  - The View in Databases

Kind Of Component : Adapters



```
package MV-RDB_Adapter
  context Class
    inv : self.name = self.table.name
    inv : self.attribute.column = self.table.column->reject(isPK or isFK)
    inv : self.state->isEmpty
  context Attribute
    inv : self.name = self.attribute.name
endpackage
```

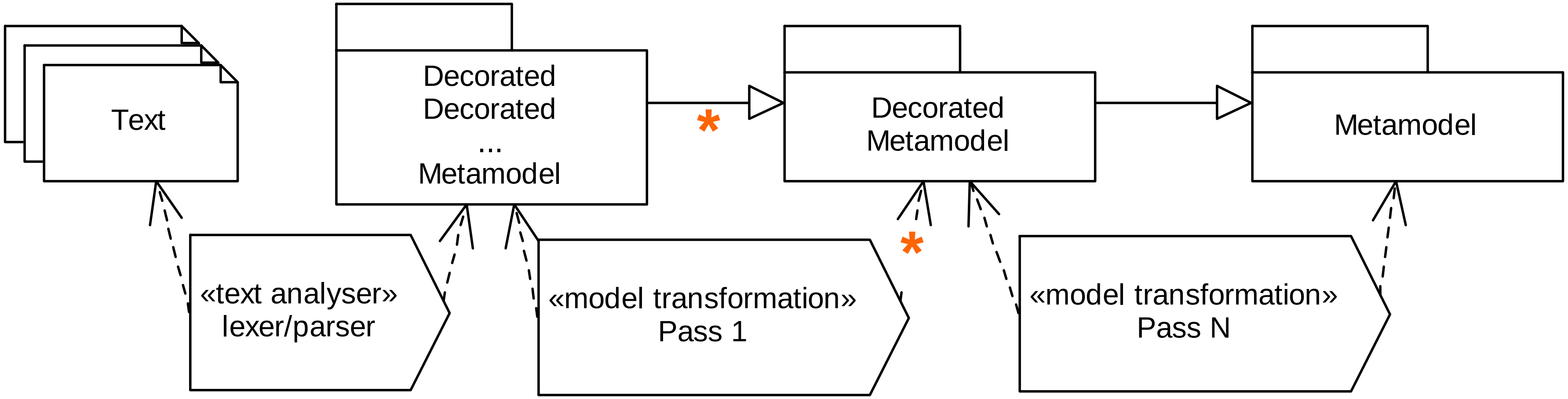
Kind Of Component : Adapters



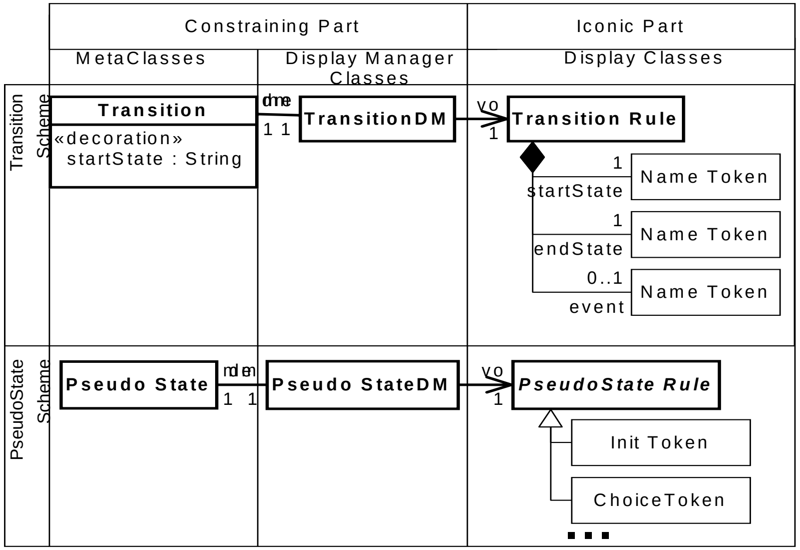
Kind Of Component : Parser

- Text (Tokens) => Model
- Often requires Several Passes

- Idea
  - First pass creates a model
  - Each pass has its model

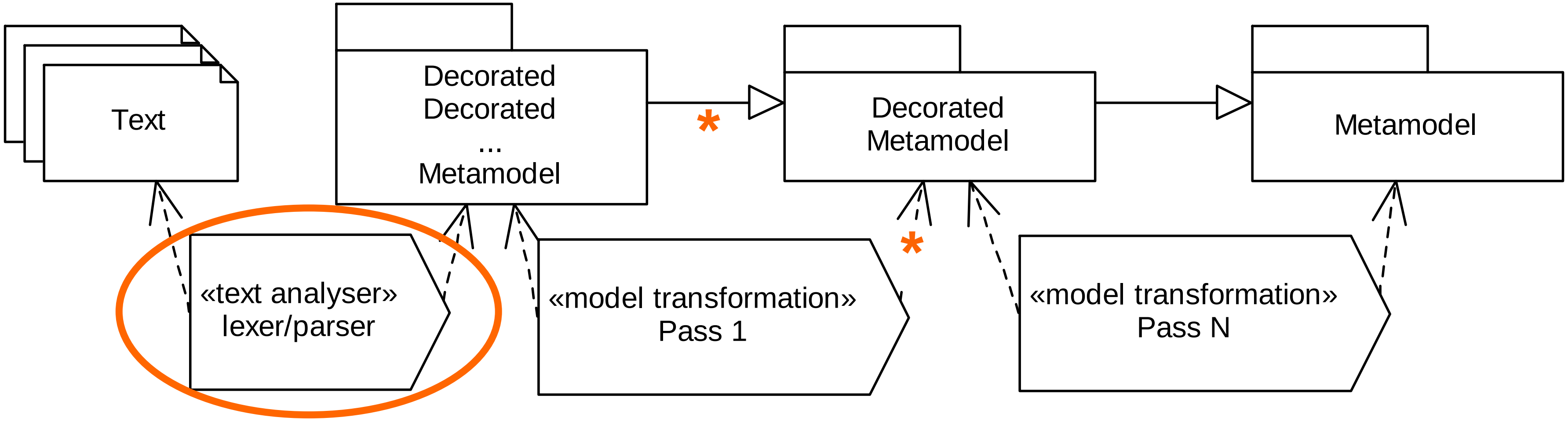


Kind Of Component : Parser



Text Analyser # Editor ?

- Text templates => (decorated) Model
- Constraints


Text
Decorated
Decorated
...
Metamodel
*
Decorated Metamodel
Metamodel
*
«text analyser» lexer/parser
«model transformation» Pass 1
«model transformation» Pass N

Other Kinds of Components

- Model Transformations
  - Compilers
  - Metrics
  - Derivation
    - Product Lines
  - …
- Interpreters
  - Requires an execution environment
  - Transformation to “Semantically-Rich” Metamodels
    - KerMeta / adapted Action Semantics / COOPN / …
    - Requires Type Translations !
- Code Generators
  - An RFP at the OMG
  - Related to **Text Analyser** construction(?)

Contents

- Language Engineering
- Language Components
- Identified Kinds of Components
- Tuning
- Outlook

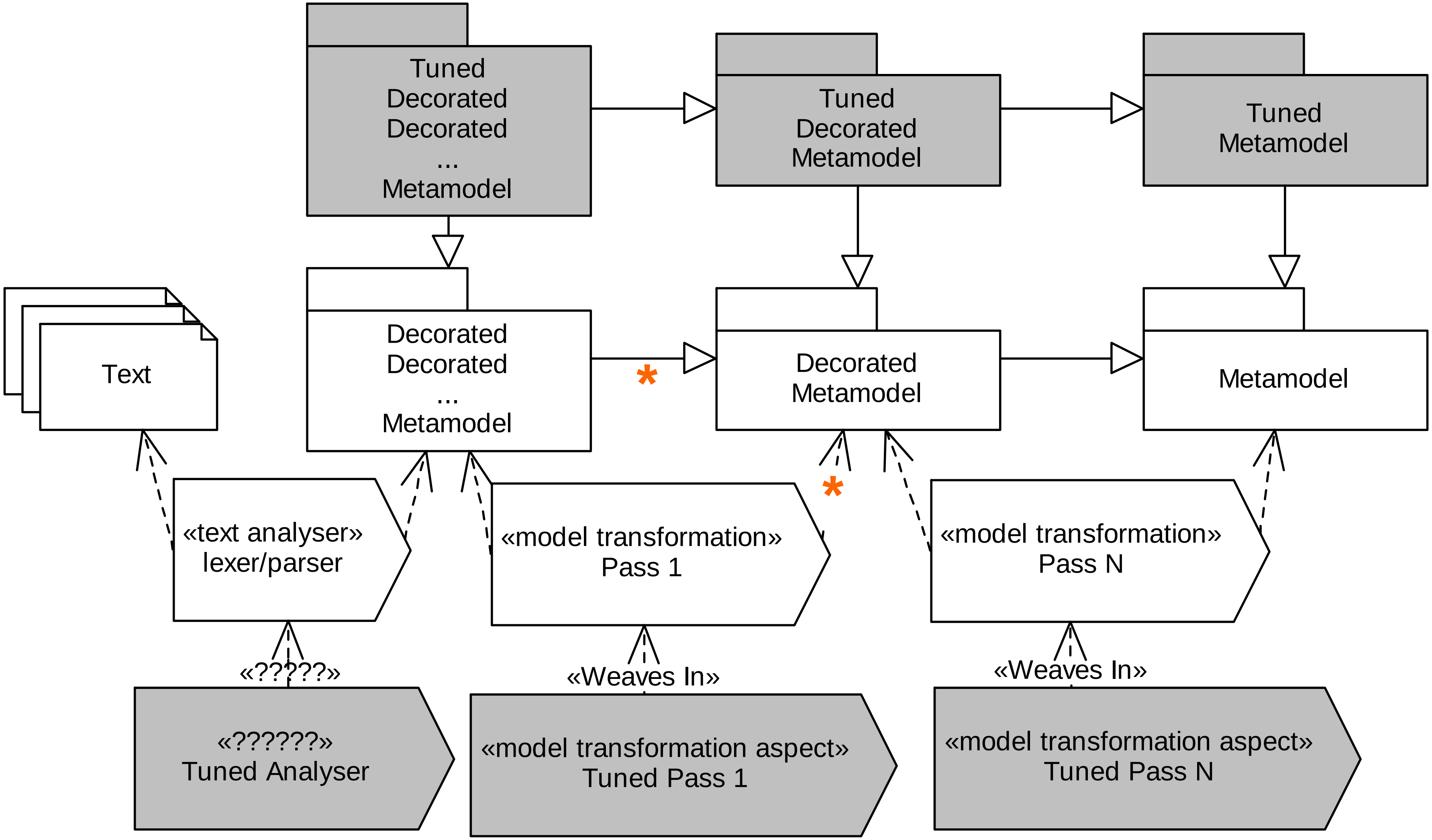
Tuning Components

- Adapt an Almost Solution
  - OCL extended to support Temporal Constraints
  - OCL extended to Fondue
  - Action Semantics extended to Web Application Engineering

- Extend Structure
- Add / Change Behaviour of Components

- Tuning Depends on the Kind of Component

Tuning Components


Tuned Decorated Decorated ... Metamodel
Tuned Decorated Metamodel
Tuned Metamodel
Text
Decorated Decorated ... Metamodel
*
Decorated Metamodel
Metamodel
*
«text analyser» lexer/parser
«model transformation» Pass 1
«model transformation» Pass N
«?????»
«Weaves In»
«Weaves In»
«??????» Tuned Analyser
«model transformation aspect» Tuned Pass 1
«model transformation aspect» Tuned Pass N

Outlook

Validate the Complete Approach

- Implement Graphical Concrete Syntax Rendering
  - Semester project completed
- Realize Concrete Parser
  - OCL is under development (thanks to Amit)
- Improve / Validate Aspects for Tuning
  - Generic Aspect Weaver as a Language Component
- Study Refinement Alternatives
  - Overloading allInstances + delete + navigation
  - Event-based

Language Modules

- Components:
  - Reusable Parts
  - Define their Vision of the Environment
    - Required Interfaces
  - Solution for a Problem
    - Provided Interfaces

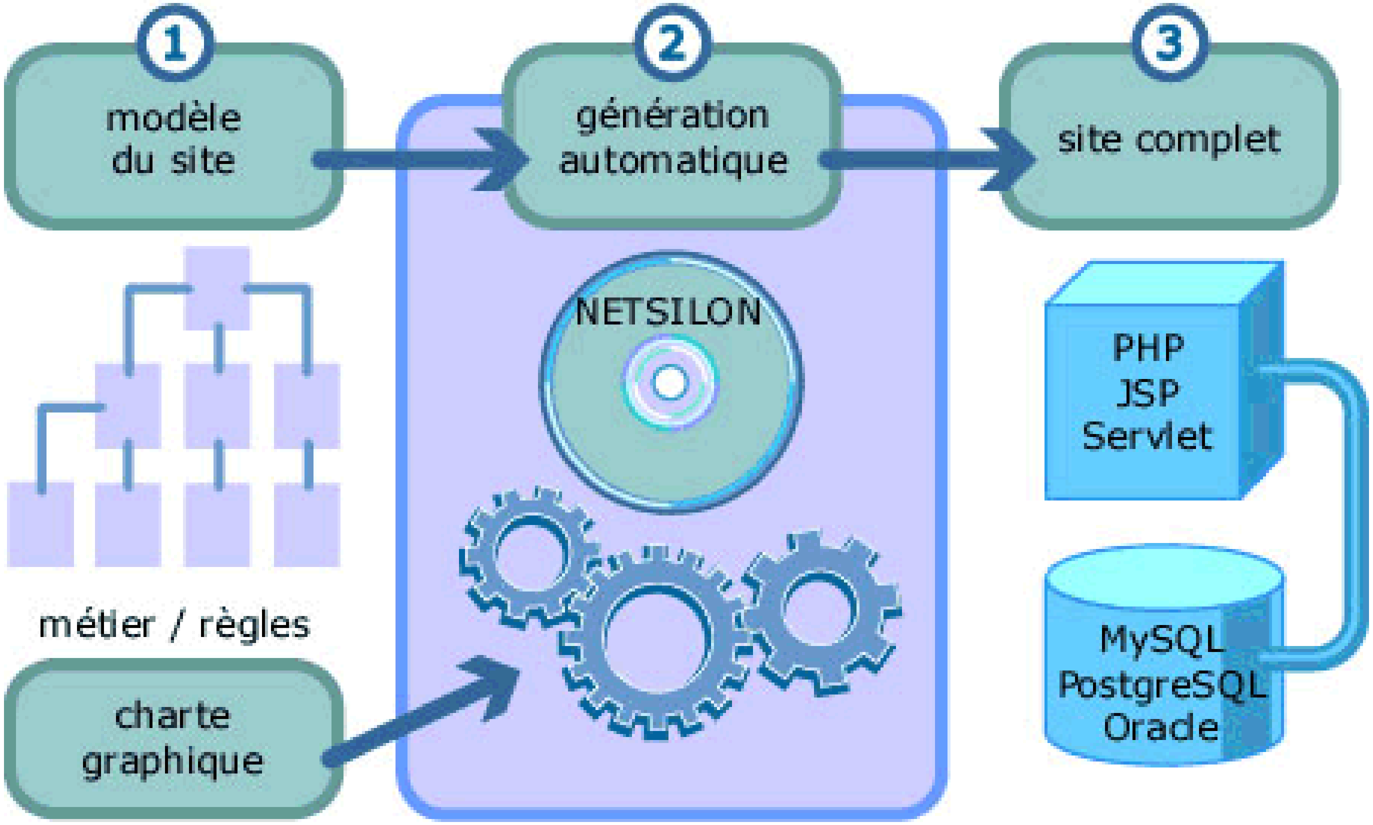
- Language Modules
  - Reusable Parts
  - Define their Vision of the Extended Language
    - Required Model
  - Solution for a Problem
    - E.g. OCL

- Language Modules
  - Can solve Semantics Level
    - Interpreter (for “Semantically-Rich” Languages)
    - Model Transformation (to “Semantically-Rich Language”)
  - Can solve Concepts Level
    - New Concepts
    - Independent from the rest of the language
      - E.g. OCL module
  - Can Solve Concrete Syntax level

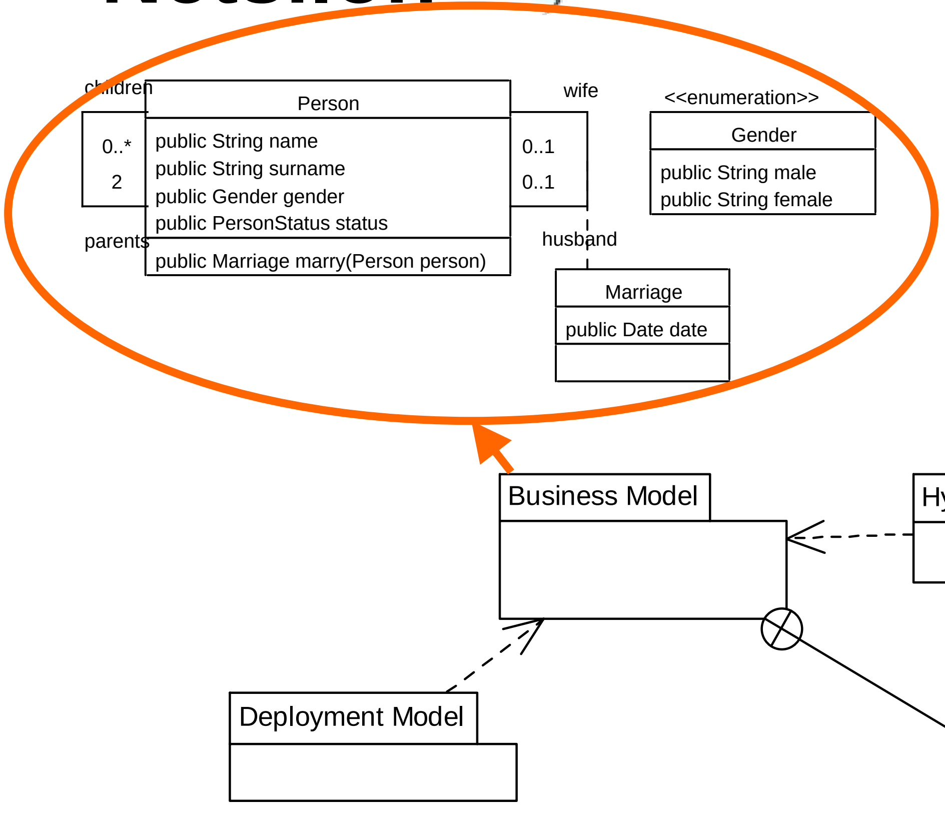
The Netsilon Example

Netsilon

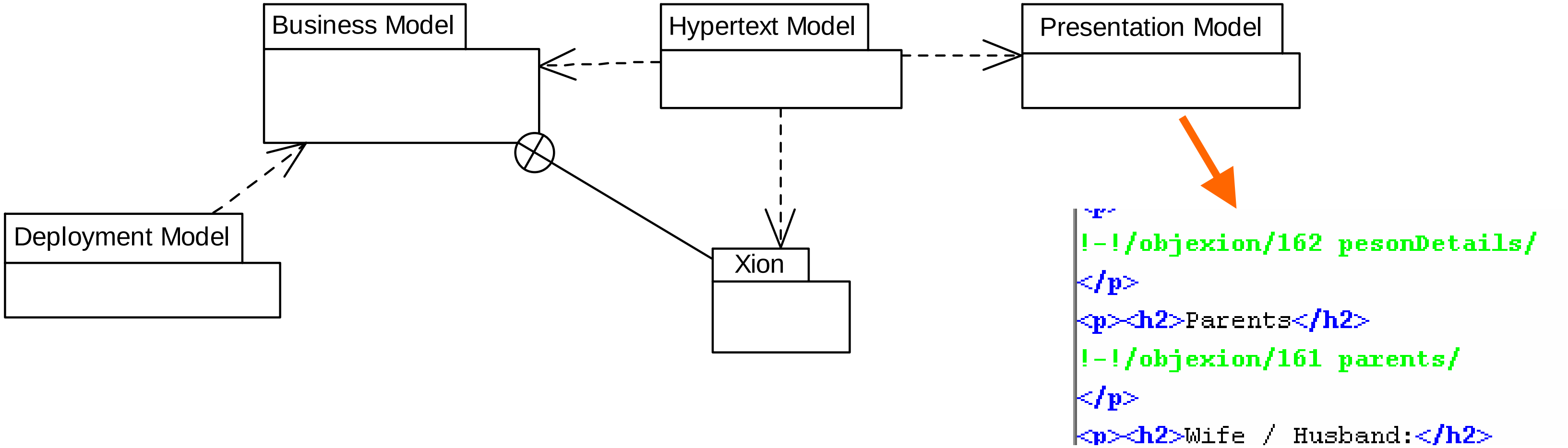
- DSL for Web Application Engineering
- Includes Multi-Platform Code Generator



Netsilon

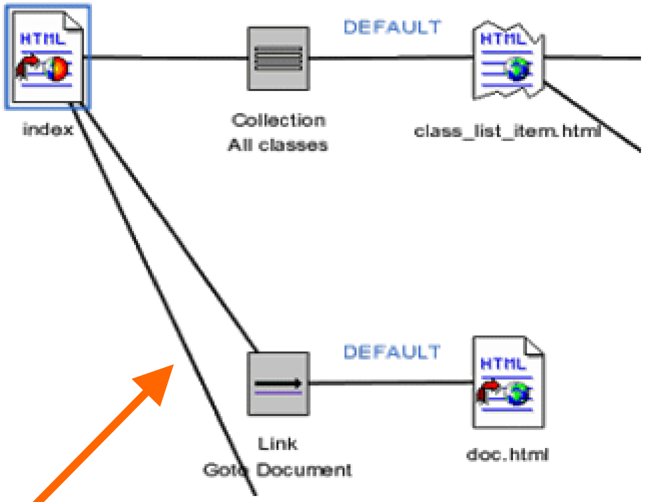

children
Person
public String name
public String surname
public Gender gender
public PersonStatus status
public Marriage marry(Person person)
0..*
2
parents
wife
0..1
0..1
husband
<<enumeration>>
Gender
public String male
public String female
Marriage
public Date date
Business Model
Hypertext Model
Presentation Model
Deployment Model
Xion

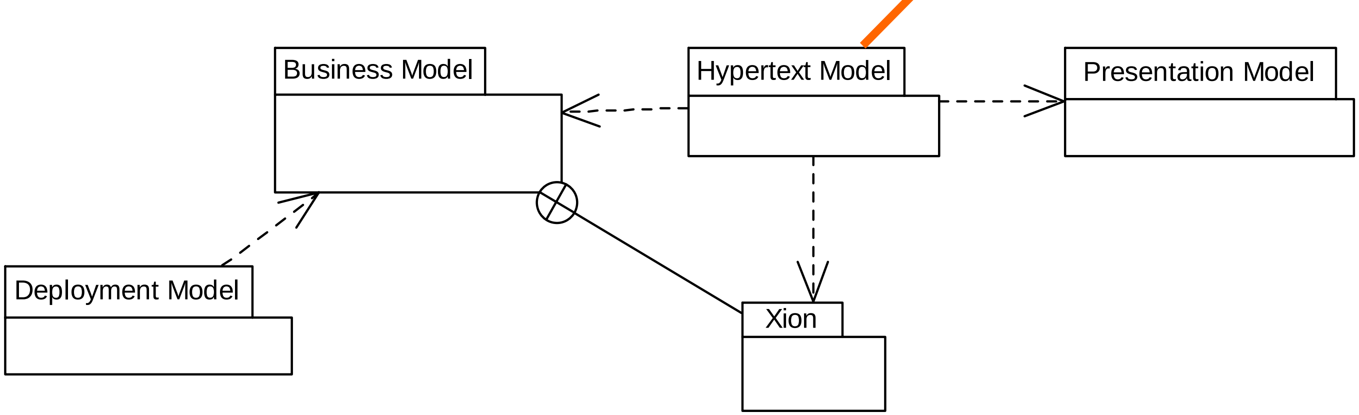
Netsilon



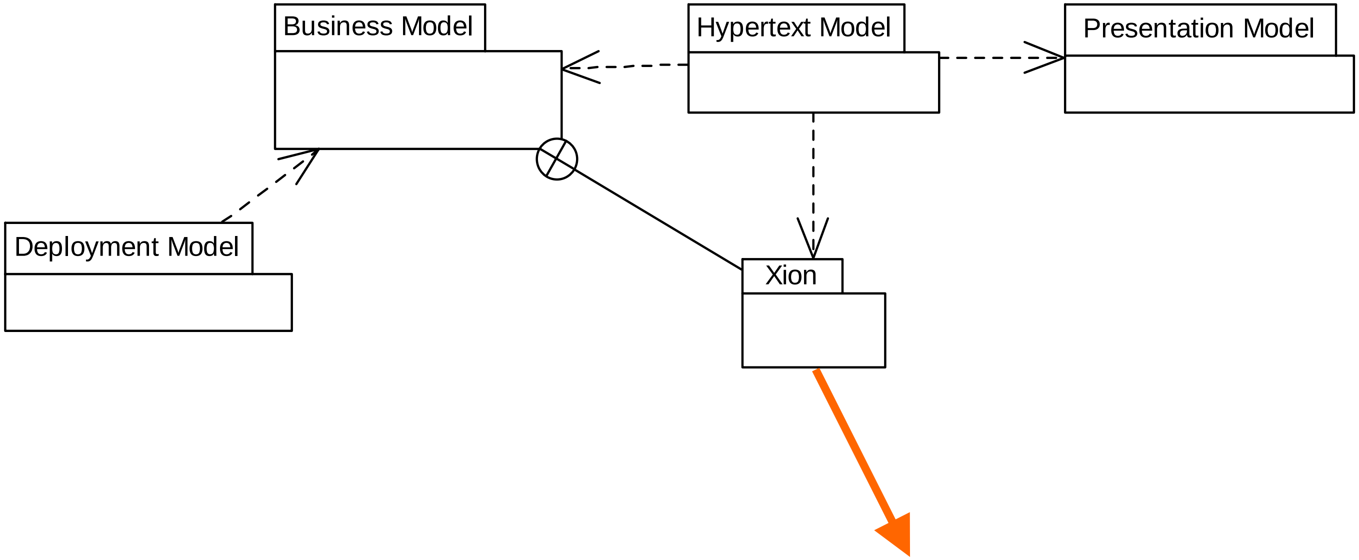
```
!-!/objexion/162 pesonDetails/
</p>
<p><h2>Parents</h2>
!-!/objexion/161 parents/
</p>
<p><h2>Wife / Husband:</h2>
```

Netsilon


DEFAULT
HTML
index
Collection
All classes
class_list_item.html
DEFAULT
HTML
Link
Goto Document
doc.html

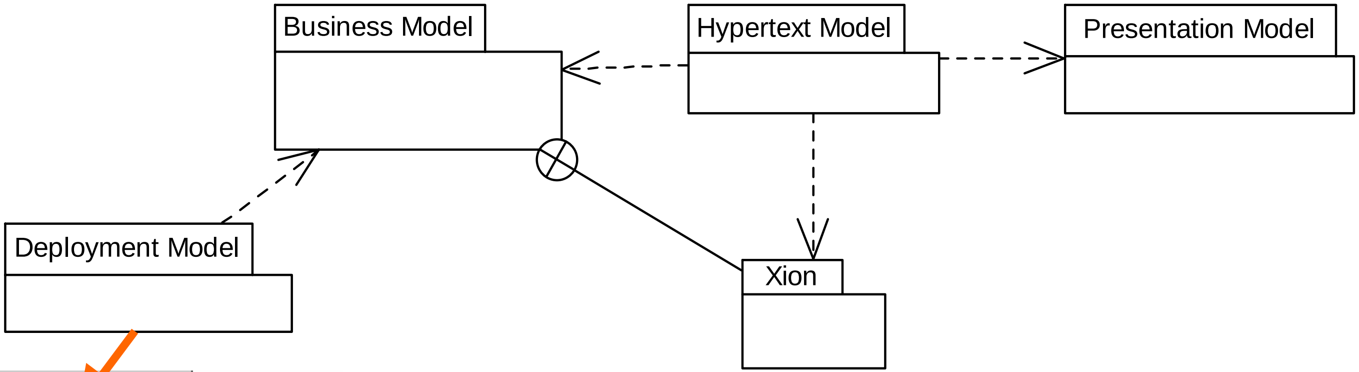

Business Model
Hypertext Model
Presentation Model
Deployment Model
Xion

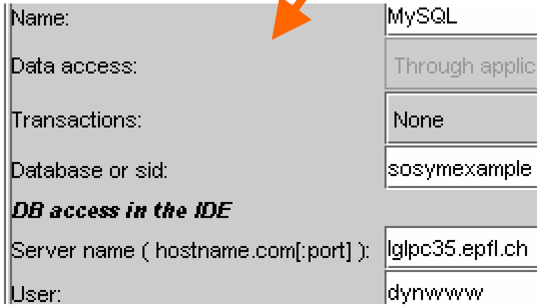
Netsilon



person.parents.children->asSet()->excluding(person)
->select(p : p.gender == #female)->sortedBy(p : p.name)

Netsilon



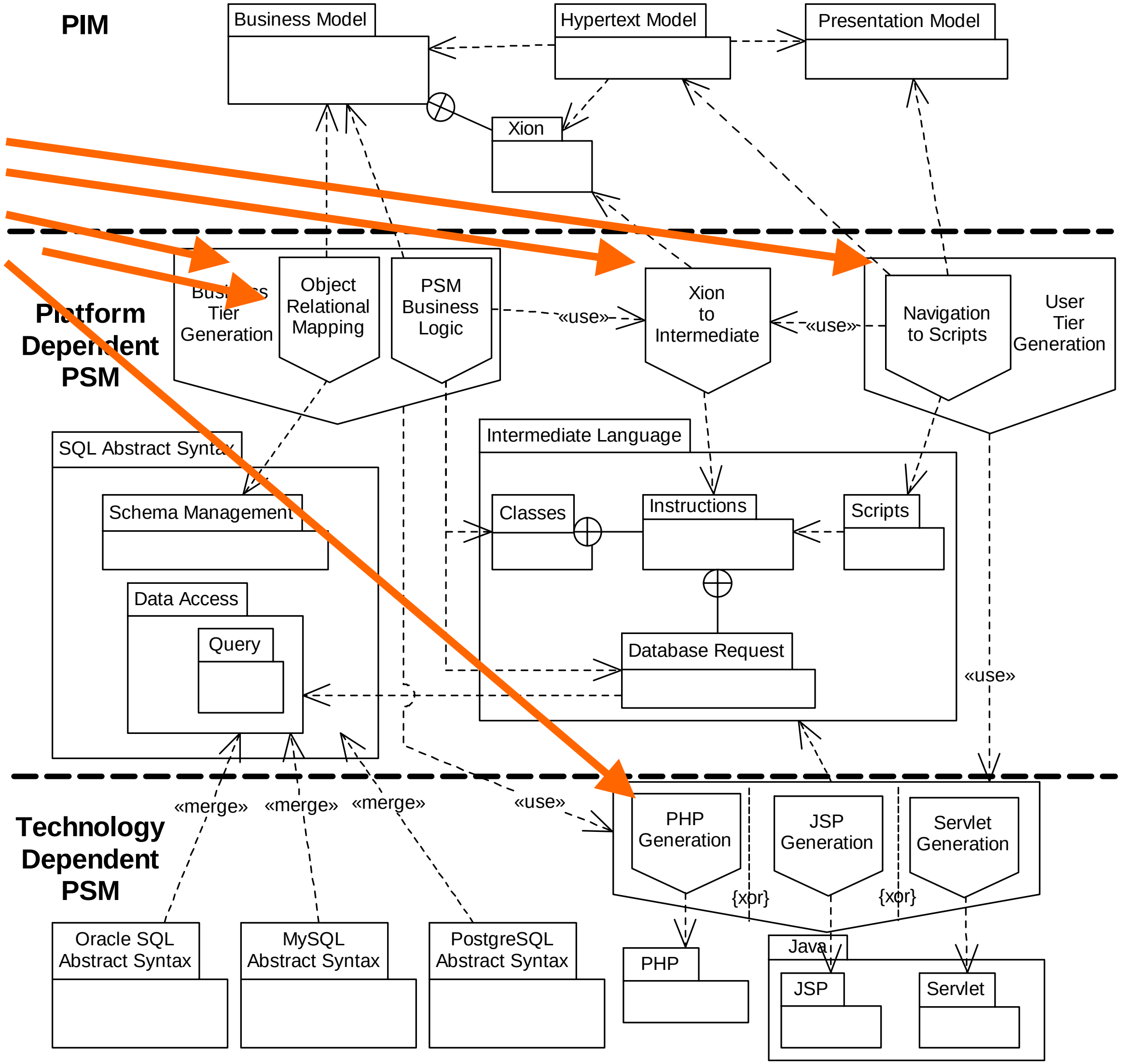


Name:	MySQL
Data access:	Through applic
Transactions:	None
Database or sid:	sosymexample
DB access in the IDE	
Server name (hostname.com[:port]):	lglpc35.epfl.ch
User:	dynwwww

Netsilon : Code Generator

Model Transformations

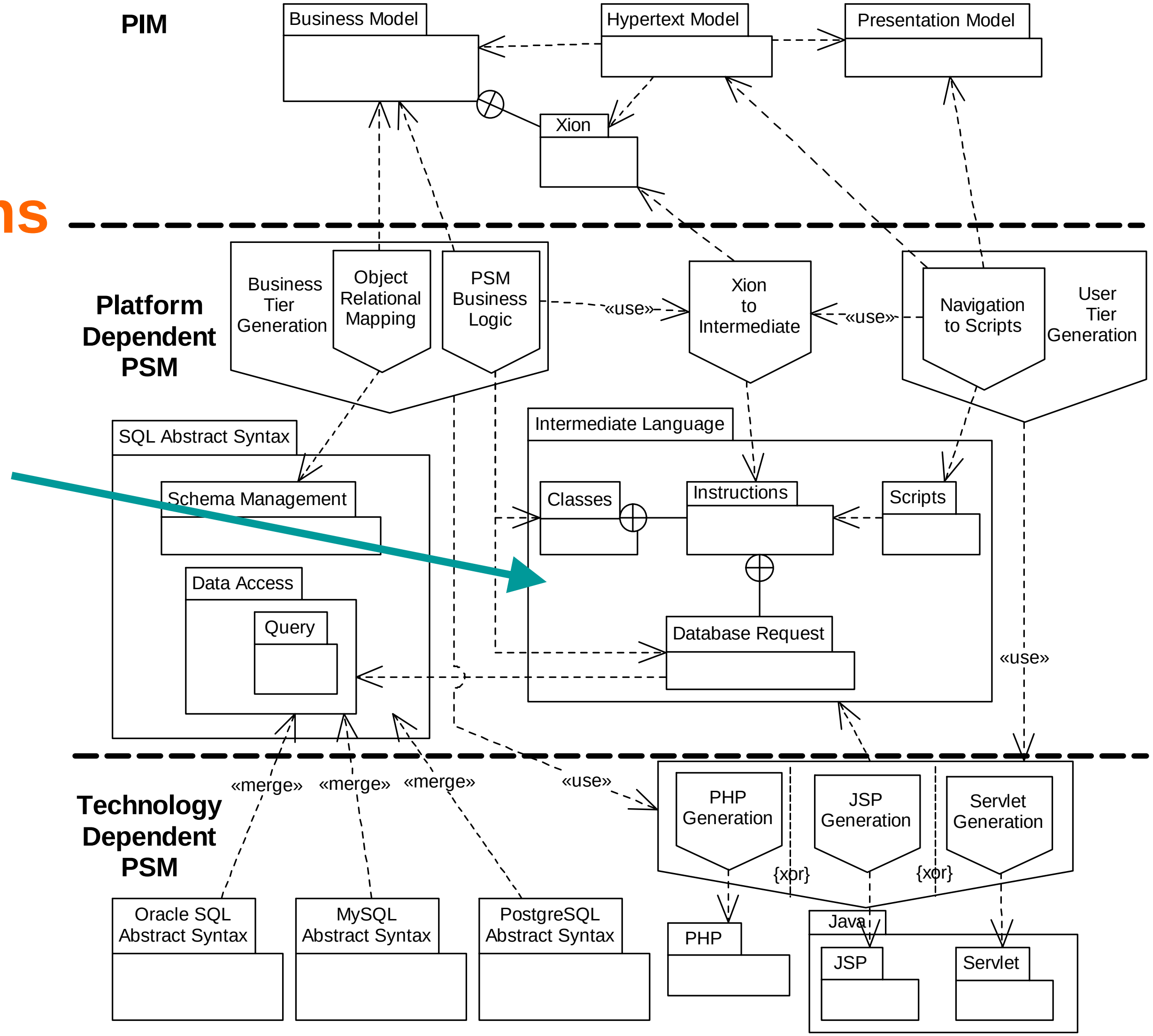
- Use
- Composition
- Selection



Netsilon : Code Generator

Model Transformations

Intermediate Language



PIM

Business Model

Hypertext Model

Presentation Model

Xion

Platform Dependent PSM

Business Tier Generation

Object Relational Mapping

PSM Business Logic

Xion to Intermediate

Navigation to Scripts

User Tier Generation

«use»

«use»

SQL Abstract Syntax

Schema Management

Data Access

Query

Intermediate Language

Classes

Instructions

Scripts

Database Request

«use»

Technology Dependent PSM

«merge»

«merge»

«merge»

«use»

PHP Generation

JSP Generation

Servlet Generation

{xor}

{xor}

Oracle SQL Abstract Syntax

MySQL Abstract Syntax

PostgreSQL Abstract Syntax

PHP

Java

JSP

Servlet

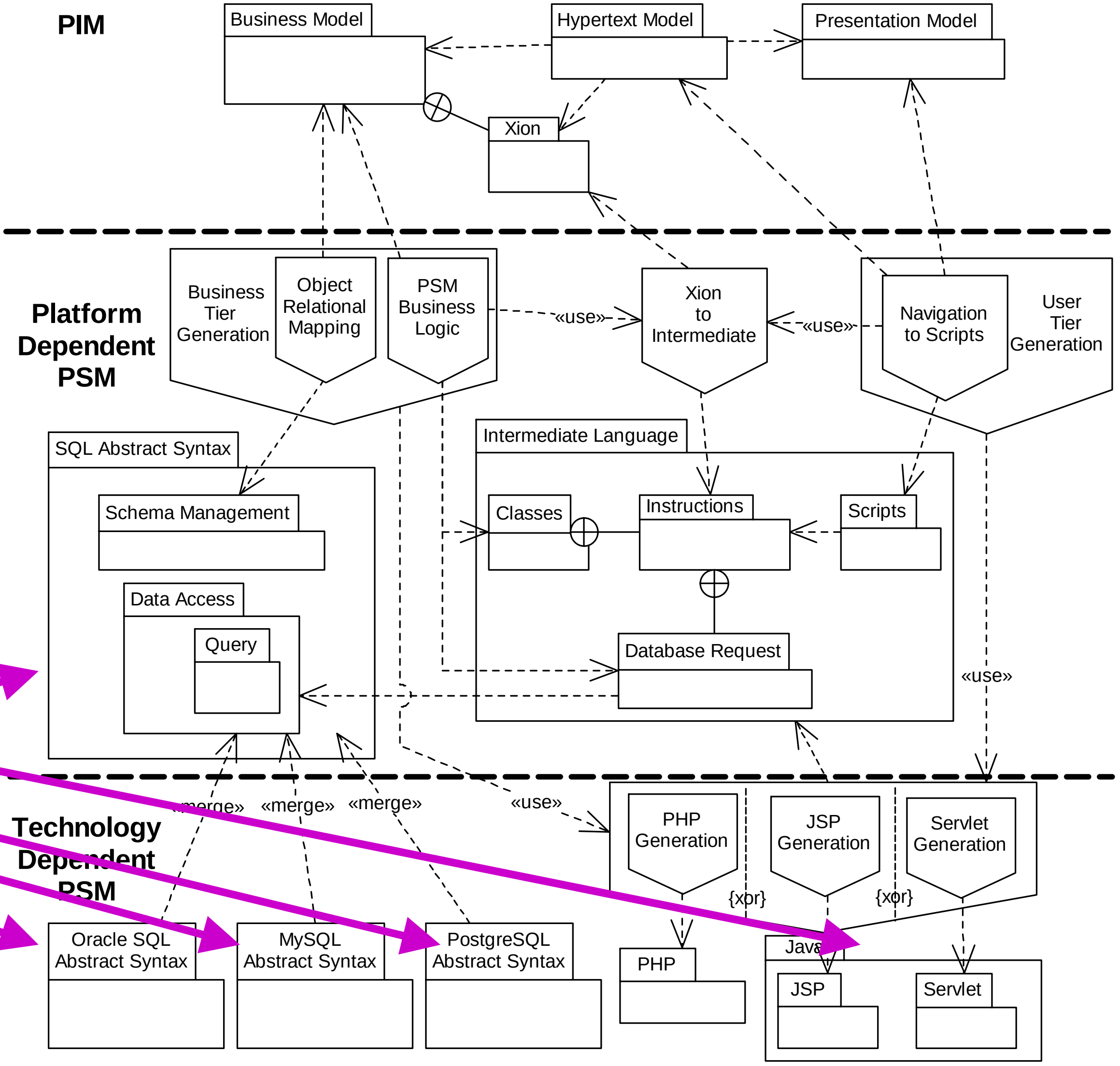
Netsilon : Code Generator

Model Transformations

Intermediate Language

Target Models

- Composition
- Refinement

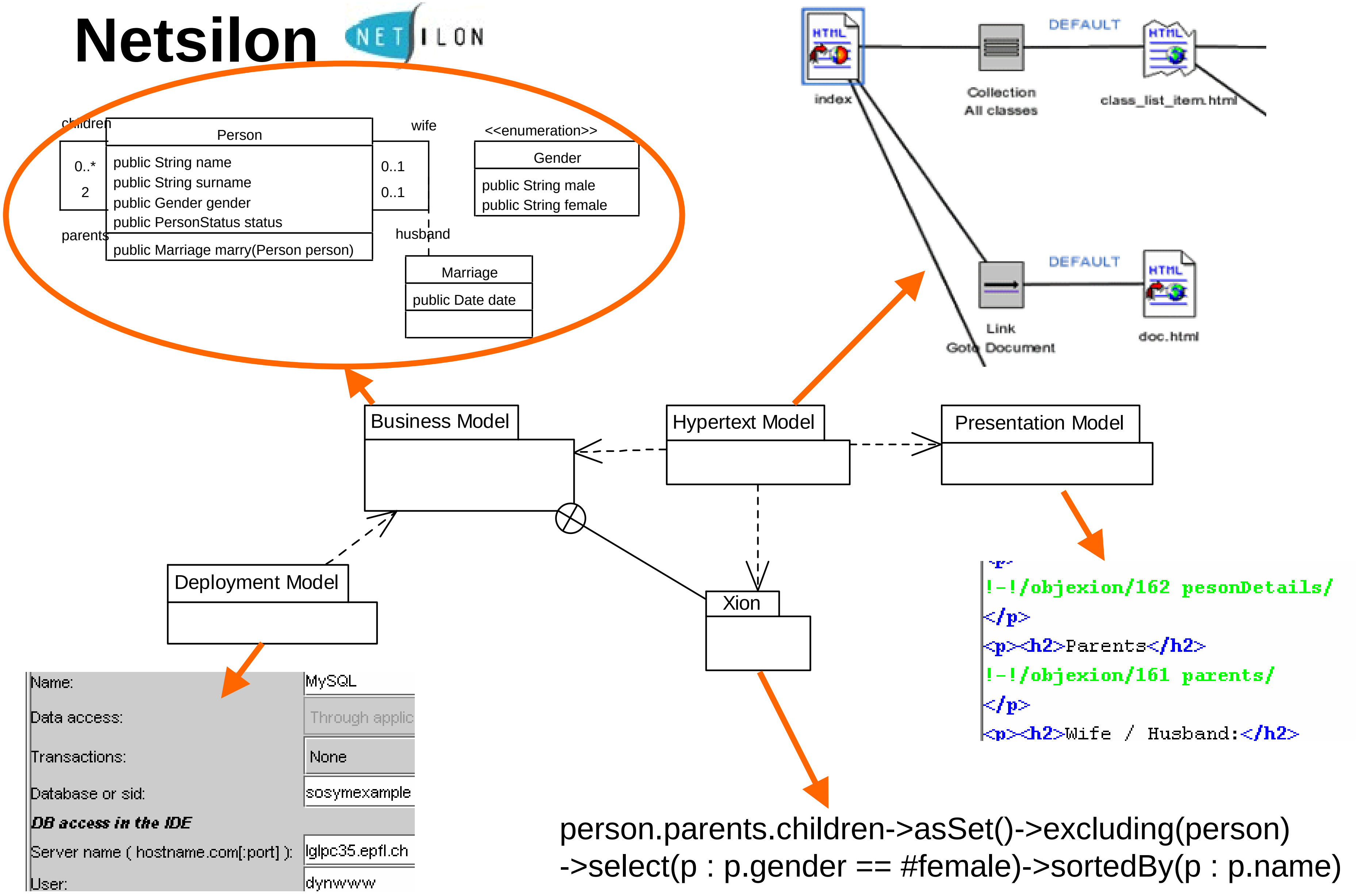


Netsilon: Realization

- Concrete Syntax
  - *Text* : Modified Text Editor
    + “XionToIntermediate” transformation
  - *Graph* : Java Library (Tigris GEF) – modified UML CASE
    + Hand-made panels
  - *Properties* : Swing
- Abstract Syntax
  - Java Classes
  - *Intermediate Language* : XML
- Semantics (i.e. Transformations and Code Generation)
  - Java (extended visitor pattern + factory pattern) + REGEXP
  - XSLT

- **A Lot of Code !**
- **No other Language Reused (OCL / UML)**
- **Nothing reusable !**

Netsilon



```
<p>
!-!/objexion/162 pesonDetails/
</p>
<p><h2>Parents</h2>
!-!/objexion/161 parents/
</p>
<p><h2>Wife / Husband:</h2>
```

Name:	MySQL
Data access:	Through applic
Transactions:	None
Database or sid:	sosymexample
DB access in the IDE	
Server name (hostname.com[:port]):	lglpc35.epfl.ch
User:	dynwww

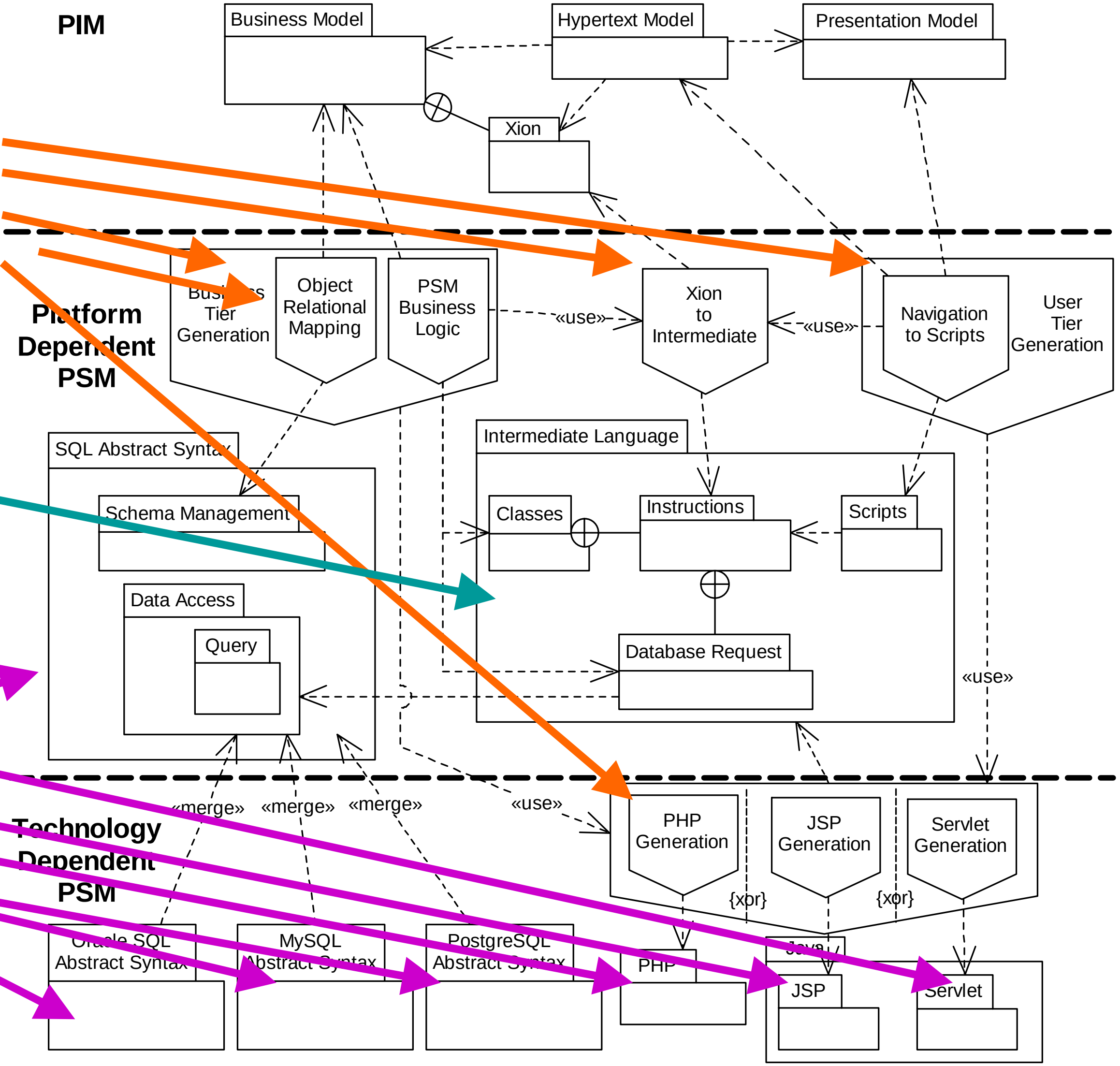
person.parents.children->asSet()->excluding(person)
->select(p : p.gender == #female)->sortedBy(p : p.name)

Netsilon : Code Generator

Model Transformations

Intermediate Language

Target Models


PIM
Business Model
Hypertext Model
Presentation Model
Xion
Platform Dependent PSM
Business Tier Generation
Object Relational Mapping
PSM Business Logic
«use»
Xion to Intermediate
«use»
Navigation to Scripts
User Tier Generation
SQL Abstract Syntax
Schema Management
Data Access
Query
Intermediate Language
Classes
Instructions
Scripts
Database Request
«use»
«merge»
«merge»
«merge»
«use»
Technology Dependent PSM
PHP Generation
JSP Generation
Servlet Generation
{xor}
{xor}
Oracle SQL Abstract Syntax
MySQL Abstract Syntax
PostgreSQL Abstract Syntax
PHP
Java
JSP
Servlet