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

S1

T12

S2

S2.1

S2.2
Language Construction

Concrete Syntax  +  Abstract Syntax
Language Construction

Concrete Syntax + Abstract Syntax + Semantics

SC AST

SC Interpreter

SC AST

SC2PN Compiler

PN AST
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

- Language Composition
- Language Adaptation
- Language Tuning

«include»
An Example: Generic OCL Framework

- OCL Parser
- UML-MV Adapter
- Java-MV Adapter
- MyMM-MV Adapter
- Java Parser
- My Repository

«metamodel» OCL MM

«metamodel» Model View

Model View

UML-MV Adapter

UML MM

UML Editor

Java AS

My MM

My Repository

© F. Fondement June 10, 2004
An Example: Generic OCL Framework

- OCL Parser
- UML-MV Adapter
- UML Editor
- Java-MV Adapter
- Java Parser
- MyMM-MV Adapter
- My Repository
- OCL Interpreter
- OCL - KM Compiler

Diagram:

- OCL MM
- UML MM
- Java AS
- My MM
- KM MM
Contents

- Language Engineering
- Language Components
- Identified Kinds of Components
- Tuning
- Outlook
Kind Of Component : Repository

- Metamodel-Driven

- Already at work
  - XMI
  - JMI
  - ...

My MM

My Repository
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!
Proposal to define (graphical) Concrete Syntaxes
  - SVG templates
    - Properties
    - Behaviours
  - Mapping model
  - Consistency Constraints
    - MM vs. Representation
    - Spatial Relationships

Kind Of Component : Editors

Constraining Part

Iconic Part

- Transition
  - TransitionDM
  - SVGTransition
- Simple State
  - Simple StateDM
  - SVGSimpleState
- Pseudo State
  - Pseudo StateDM
  - SVGpseudo State
- SVGNamespace

- «Interface» GraphicalObject
  - contains()
  - connects()
  - nearby()
  - overlaps()
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

- 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

- «MTLLibrary» ModelView
- «MTLLibrary» MOF20Adapter
- «MTLLibrary» Core20Adapter
- «MTLLibrary» OCLAbstractSyntax
- «model» uml20 {refines core20} 1
- «model» mof20 {refines core20} 1
- «model» OCL4Fondue
Kind Of Component : Parser

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

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

Diagram:
- Text
- Decorated ... Metamodel
  - «text analyser» lex/par
  - «model transformation» Pass 1
- Decorated Metamodel
- Metamodel
  - «model transformation» Pass N
Kind Of Component : Parser

Text Analyser # Editor ?
- Text templates => (decorated) Model
- Constraints

### Constrain Part

<table>
<thead>
<tr>
<th>MetaClasses</th>
<th>Display Manager Classes</th>
<th>Iconic Part</th>
<th>Display Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>TransitionDM</td>
<td>TransitionRule</td>
<td></td>
</tr>
<tr>
<td>startState</td>
<td>String</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pseudo State</th>
<th>Pseudo StateDM</th>
<th>PseudoStateRule</th>
<th>Iconic Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pseudo State</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pseudo State Rule:
- Init Token
- ChoiceToken

Text Analyser:
- «text analyser»
- lexer/parser

Model Transformation:
- «model transformation»
  - Pass 1
  - Pass N

Metamodel:
- «model transformation»
- Display Classes
- Iconic Part
- MetaClasses
- Metamodel
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

- Text
  - «text analyser» lexer/parser
  - «??????» Tuned Analyser
- Tuned Decorated Decorated ... Metamodel
  - «model transformation» Pass 1
  - «Weaves In» «model transformation aspect» Tuned Pass 1
- Decorated Decorated ... Metamodel
  - «model transformation» Pass 1
  - «Weaves In» «model transformation aspect» Tuned Pass 1
- Tuned Decorated Metamodel
  - «model transformation» Pass N
  - «Weaves In» «model transformation aspect» Tuned Pass N
- Metamodel
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
- DSL for Web Application Engineering
- Includes Multi-Platform Code Generator
Netsilon

Business Model

Hypertext Model

Presentation Model

Deployment Model

Xion

Person

- public String name
- public String surname
- public Gender gender
- public PersonStatus status
- public Marriage marry(Person person)

Gender

- public String male
- public String female

Marriage

- public Date date

children

0..*

2

parents

0..1

0..1

husband

wife

<<enumeration>>

0..*

children

2

parents

0..1

husband

wife

<<enumeration>>
Netsilon

Business Model

Hypertext Model

Deployment Model

Presentation Model

Xion

<!-- !-/objexion/162 pesonDetails/ -->

<p>
</p>

<h2>Parents</h2>

<!-- !-/objexion/161 parents/ -->

<p>
</p>

<h2>Wife / Husband:</h2>
person.parents.children->asSet()->excluding(person)->select(p : p.gender == #female)->sortedBy(p : p.name)
Netsilon : Code Generator

Model Transformations
- Use
- Composition
- Selection

[Diagram showing various transformations and layers involving Business Model, Hypertext Model, Presentation Model, Xion, Intermediate Language, SQL Abstract Syntax, PIM, Platform Dependent PSM, Technology Dependent PSM, and Generation to Scripts.]
Netsilon: Code Generator

Model Transformations

Intermediate Language

Platform Dependent PSM

- Business Model
- Hypertext Model
- Presentation Model

PSM to Intermediate

- Xion
- SQL Abstract Syntax
- Schema Management
- Data Access
- Query
- Database Request

Intermediate Language

- Classes
- Instructions
- Scripts
- Navigation to Scripts

Technology Dependent PSM

- Oracle SQL Abstract Syntax
- MySQL Abstract Syntax
- PostgreSQL Abstract Syntax
- PHP Generation
- JSP Generation
- Servlet Generation

User Tier Generation

- PHP
- JSP
- Servlet

Object Relational Mapping

- PSML to Business Logic

User Tier Generation

- Navigation to Scripts

Hypertext Model

- Xion

Business Model

- Xion

Presentation Model

- Xion

Business Logic

- Xion

Object Relational Mapping

- Xion

Database Request

- Xion

Query

- Xion

Data Access

- Xion

Intermediate Language

- Xion

Platform Dependent PSM

- Xion

PSM Dependent PSM

- Xion

User Tier Generation

- Xion

Technology Dependent PSM

- Xion

Oracle SQL Abstract Syntax

- Xion

MySQL Abstract Syntax

- Xion

PostgreSQL Abstract Syntax

- Xion

PHP Generation

- Xion

JSP Generation

- Xion

Servlet Generation

- Xion

© F. Fondement

June 10, 2004
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!
public String name
public String surname
public Gender gender
public PersonStatus status
public Marriage marry(Person person)

<<enumeration>>
public String male
public String female

-->

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

<h2>Parents</h2>

<h2>Wife / Husband:</h2>

-->

MySQL

None

sosymexample

lypc35.epfl.ch

dynwww
Netsilon: Code Generator

Model Transformations

Intermediate Language

Target Models