Dynamism in Refactoring Construction and Evolution
A Solution Based on XML and Reflection

Authors:
Raúl Marticorena
Yania Crespo

rmartico@ubu.es
yania@infor.uva.es
Outline

- Introduction
- MOON as Metamodell
- Running Refactorings
- Refactoring Construction and Evolution
- Current Work
- Conclusions and Future Work
Dynamism in Refactoring Construction and Evolution

Introduction

Refactoring [Fowler, 2000]

“Process of changing a software system in such a way that it does not alter external behavior of the code yet improve its internal structure”

Example: Add Parameter

Open Research Trends

- Define new refactorings
- Identify code defects *(Bad Code Smells)*
- Apply refactorings
- Tool support with certain language independence
- etc.

Porto, Portugal, july 2008
Introduction

- **Language Representation**
  - Abstract Syntax Tree (AST)
  - Database
  - Logical predicates
  - XML
  - **Metamodel**
    - Language independence

- **Goals**
  - Aided refactoring construction
  - Refactoring evolution with low effort
  - Running refactorings with certain language independence
  - Reuse elements

Porto, Portugal, July 2008
MOON as Metamodel

MOON [Crespo 2000]

- Minimal Object-Oriented Notation
  - abstractions for refactoring
  - 50 classes

- Storing:
  - Classes
  - Relationships
  - Variants on the type system
  - Entities
    - Concepts in source code with type
      - self reference, super reference, local variable, method
        formal argument, class attribute and function result
  - Expresssions
  - Instructions
    - creation, assignment, call and compound instructions
MOON as Metamodel

1. General concepts: defined and implemented on MOON

2. Extensible:
   - Defined on MOON
   - Implemented on concrete language (framework instantiation)

3. Particular: defined and implemented on a concrete language

---

Porto, Portugal, July 2008
Running Refactorings

- Framework engine
  - Actions & Queries

```
<abstract>> Predicate
isValid
defined by
0..*
```

```
<abstract>> Function
getValue()
getCollection()
```

```
<abstract>> Refactoring
name
description
motivation
```

```
runActions()
undoActions()
validatePre()
validate(Post)
addPre()
addOPost()
run()
```

```
Action
run()
undo()
```

Refactoring Core

Porto, Portugal, July 2008
Running Refactorings

- Frameworks as solution
  - Repositories with actions & queries

Diagram:
- MOON
- JavaMOON
- MOON Refactoring Repository
- Java Refactoring Repository
- Refactoring Core
From hand coded refactorings ...

To aided construction

Dynamism in Refactoring Construction and Evolution

- Introduction
- MOON as Metamodel
- Running Refactorings
- Refactoring Construction and Evolution
- Current Work
- Conclusions and Future Work

Porto, Portugal, July 2008
Refactoring as XML file

```
<xml>
  <refactoring>
    <input>
    ... 
    </input>
    <actions>
      <!-- Actions here -->
    </actions>
  </refactoring>
</xml>
```

Not hand coded
Not compiled
Saved as XML
How can we run a refactoring from XML?

Reusing the refactoring engine

- Reflection as basic mechanism

```xml
<xml>
  <refactoring>
    <input> ... </input>
  </refactoring>
</xml>
```
Dynamism in Refactoring Construction and Evolution

Refactoring Construction and Evolution

- Introduction
- MOON as Metamodel
- Running Refactorings
- Refactoring Construction and Evolution
- Current Work
- Conclusions and Future Work

12 of 16

Loading elements

MOON Repository

Action

Predicate

Function

Language Repository

Action

Predicate

Function

XML files

DynamicEngine

Engine

Refactoring

RefactoringXML

Porto, Portugal, July 2008
Problem

- Solution is prone to failures building the XML file manually
- Wizard to assist the construction
  - Using reflection
  - Showing inputs, pre/postconditions (queries) and actions
Current Work

First prototype (static solution)

- Hand coded refactorings
  - Add Parameter (275)
  - Rename Method (273)
  - Rename Class
  - Rename Parameter
  - Move Field (146)
  - Move Method (142)
  - Remove Parameter
  - Refactorings on generic features...
    - Specialize Bound S
    - Etc.

Second phase:

- XML solution with wizard
- Reflection mechanism as solution

Current phase:

- Improving XML format
- Developing an Eclipse plug-in
  - Partial language support
  - Currently working with example codes
Conclusions and Future Work

- Users can construct and evolve refactorings
  - Not re-compilation
  - Aided modification using wizard

- Reuse refactorings
  - As composition of items / elements

- New refactorings to measure the reuse level

- Apply refactorings to migrate libraries with new features
Thank you very much

Authors: Raúl Marticorena
Yania Crespo

rmartico@ubu.es
yania@infor.uva.es