Error Detection by Refactoring Reconstruction

International Workshop on Mining Software Repositories
May 17, 2005
Saint Louis

Carsten Görg
Saarland University
goerg@cs.uni-sb.de

Peter Weißgerber
Catholic University Eichstätt
peter.weissgerber@ku-eichstaett.de
Refactoring

„Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.“

(Martin Fowler)
Effects of Local Refactorings

Renaming Methods to Change Event Handling

```java
import java.awt.event.MouseAdapter;
```
Outline

Reconstructing refactorings

Checking consistency

Case studies on open source projects
Detecting and Visualizing Refactorings from Software Archive

Search for refactorings on the level of classes and methods

Structural refactorings
- Move Class
- Move Method (Pull up / Push down)

Local refactorings
- Rename Method
- Add/Remove Parameter To/From Method
- Hide Method

[Goerg, Weisgerber: IWPC2005]
CVS Repository

Preprocessing

Versions, Transactions, Configurations

Analyses

Refactorings, Class Hierarchy

Graph drawing

Visualization

[Zimmermann, Weißgerber: Preprocessing CVS Data for Fine-Grained Analysis, MSR2004]
Artefacts within one transaction

Parse each revision of each transaction

Identify
Classes
Full qualified name

Fields
Name and type

Methods
Name, parameter list, return type
Compare artefacts of current and previous revision added, removed, common artefacts
Uncovering refactorings

Use simple heuristics

Consequences:
Only candidates
Problem of impure and ambiguous refactorings
„Rename Method“

Method A added in new version
Method B deleted from previous version

same parameters
same return type
same method body
“Add / Remove Parameter”

Method A added in new version
Method B deleted from previous version

same name

same return type

parameter list of A is a subset of parameter list of B
Ambiguous Refactorings

It is not always possible to unambiguously identify refactorings:

Old configuration:

Class A
m(a,b)
m(b,c)

New configuration:

Class A
m(a,b,c)
Checking Consistency

For each refactoring compute list of other possible candidates

2 types of candidates
  Methods in subclasses
  Methods in sibling classes

Check if missing refactoring has been applied in a later transaction
Checking Consistency

Different types of possible errors:

Subclasses
- Interface method or abstract method is no longer implemented
- Refactored method is inherited instead of being overwritten

Sibling classes
- Wrong program behavior
Case studies – refactorings in open source projects

JEdit
6 developers
2141 Transactions
Start: September'01

Jakarta Tomcat
40 developers
5096 transactions
Start: Oktober'99
Our contribution

Approach to reconstruct refactorings from software archives

Check consistency

Case studies

  jEdit: 5 candidates for erroneous transactions
  (3 subclasses, 2 sibling classes)

  TomCat: 7 candidates for erroneous transactions
  (3 subclasses, 4 sibling classes)