Making Software Refactorings Safer

Anna Maria Eilertsen
@Sowhow

Supervised by: Anya Bagge[1] Volker Stolz [2]

[1]Inst. for Informatikk, Universitetet i Bergen

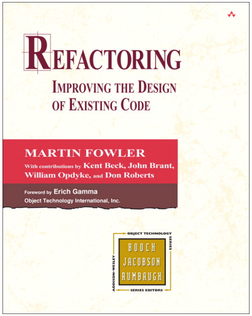
[2]Inst. for Data- og Realfag, Høgskolen i Bergen
Norway

July 12, 2016

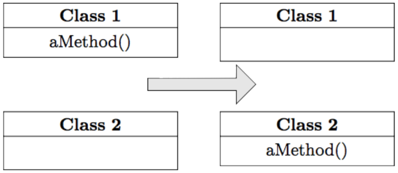
The results of this thesis has been accepted to the ISOLA[1] conference as a paper.

[1]http://www.isola-conference.org/isola2016/

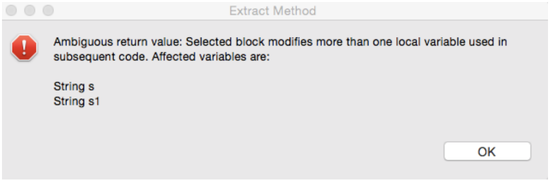
Software Refactorings


REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International, Inc.
BOOCH JACOBSON RUMBAUGH

"Behaviour preserving program transformation"



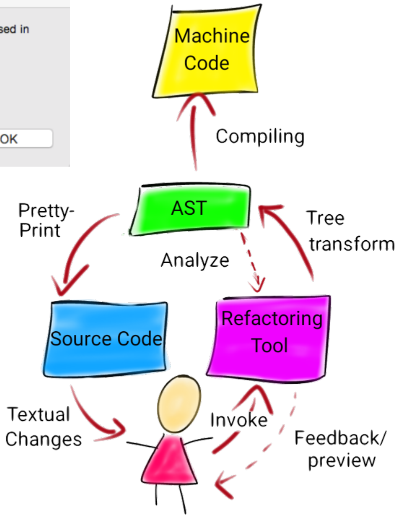
Software Refactoring Tools


Extract Method

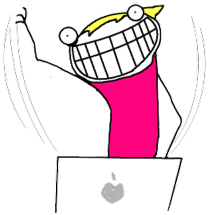
Ambiguous return value: Selected block modifies more than one local variable used in subsequent code. Affected variables are:

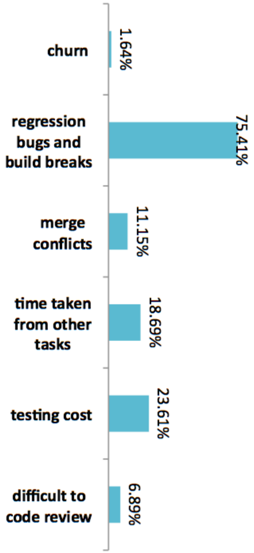
String s
String s1

OK



Unsafe Refactorings





"The primary risk is regression, mostly from misunderstanding subtle corner cases in the original code and not accounting for them in the refactored code."

– interviewee, Microsoft developer, Kim et al., 2012

Unsafe Refactoring Example

Extract Local Variable

In Java/Eclipse:

Before

```java
public void f() {
    x.n();
    setX();
    x.n();
}
```

After

```java
public void f() {
    X temp = x;
    temp.n();
    setX();
    temp.n();
}
```

An analysing problem

```
public void f() {
    X temp = x;
    temp.n();
    setX();
    temp.n();
}
```

```
x = new X();
setX(); // x = new X();
```

Solution:

```
assert temp == x;
```

*Dog art from Hyperbole and a Half

Extract Local Variable

Simplified example:

```
public class C {
    public X x = new X();
    {//initializer
       x.myC = this;
    }

    public void f(){
        x.n();
        x.m();
        x.n();
    }
}
```

```
public class X{
    public C myC;

    public void m(){
        myC.x = new X();
    }

    public void n(){
        System.out.println(
         this.hashCode());
    }
}
```

Output:

```
1735600054
21685669
```

skip example

Extract Local Variable

Refactored:

```java
public class C {
    public X x = new X();
    {//initializer
      x.myC = this;
    }

    public void f(){
        X temp = x;
        temp.n();
        temp.m();
        temp.n();
    }
}
```

```java
public class X{
    public C myC;

    public void m(){
        myC.x = new X();
    }

    public void n(){
        System.out.println(
         this.hashCode());
    }
}
```

Output:
1735600054
1735600054

Extract Local Variable

With dynamic checks:

```
public class C {
    public X x = new X();
    {//initializer
      x.myC = this;
    }
    public void f(){
        X temp = x;
        assert temp == x;
        temp.n();
        assert temp == x;
        temp.m();
        assert temp == x;
        temp.n();
```

```
public class X{
    public C myC;

    public void m(){
        myC.x = new X();
    }

    public void n(){
        System.out.println(
         this.hashCode());
    }
}
```

Output:

```
1735600054
Exception in thread "main" java.lang.AssertionError
```

Extract And Move Method

A similar problem:

```
public class C {
    public X x = new X();
    {//initializer
      x.myC = this;
    }
    public void f(){
        x.bar(this);
    }
}
```

```
public class X{
   ...
    void bar(C c){
        this.n();
        assert this == c.x;
        this.m();
        assert this == c.x;
        this.n();
    }
}
```

Similar how?
Evaluate `x` once.
Refer to that value by `this`
Substitute every occurrence of `x` with `this`

Experiment: Case study

Case: Eclipse JDT UI source code

Experiment:

- Execute our modified refactorings on Eclipse JDT UI project
- Run Eclipse test suite
- Look for triggered asserts
- Profit!!

Need custom automated refactoring tool.

Experiment: Results

	Extract Local Variable	Extract and Move Method
Executed refactorings	4538	755
Total number of asserts	7665	610
Resulting compile errors ✗	0	14
Successful Tests ✓	2392	2151
Unsuccessful Tests ✗✗	4	245
Asserts triggered	2 / 136*	0

* 136 instances of the same 2 assert statements

Discussion

	Extract Local Variable	Extract and Move Method
Executed refactorings	4538	755
Total number of asserts	7665	610
Resulting compile errors ✗	0	14
Successful Tests ✓	2392	2151
Unsuccessful Tests ✗✗	4	245
Asserts triggered	2 / 136	0

Take-away and questions:

- Dynamic preconditions can be useful!
- Assert statements are incomplete.
- Show or hide the asserts from the programmer?
- Is reference equivalence too strict?

Thank you!

* Art with face is from Hyperbole and a Half

Experiment: Development

Eclipse refactoring plug-in

- Modify Eclipse's refactorings to introduce asserts
  - Extract Local Variable
  - Extract And Move Method
- Automate refactoring process
  - Execute on Java project
  - One refactoring per method
- Custom heuristic for finding refactoring targets