

GPCE'04 Software Transformation Systems Workshop  
(Vancouver, Canada, November 24th, 2004)

# Transforming Object-Oriented Programs into Structurally Reusable Components

---

Hironori Washizaki

National Institute of Informatics, Tokyo, Japan

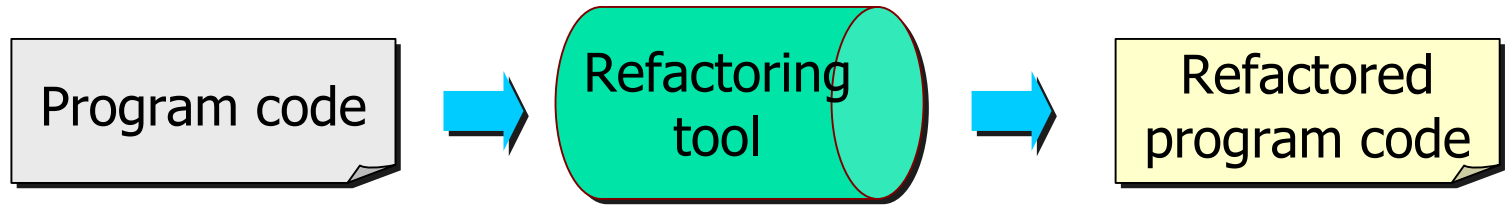
Yoshiaki Fukazawa

Waseda University, Tokyo, Japan

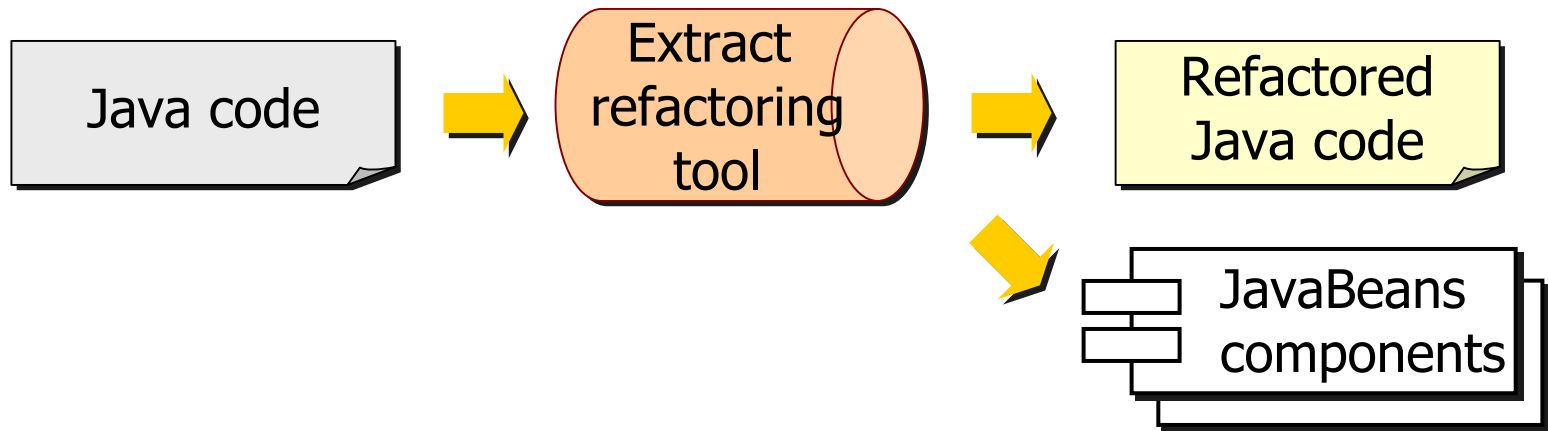
<http://www.washizaki.net/>

# Refactoring as Transformation

- Refactoring



- Extract Component Refactoring



# Class Relation Graph (CRG)

- Multigraph that represents relations among OO classes

```
public abstract class A {}
```

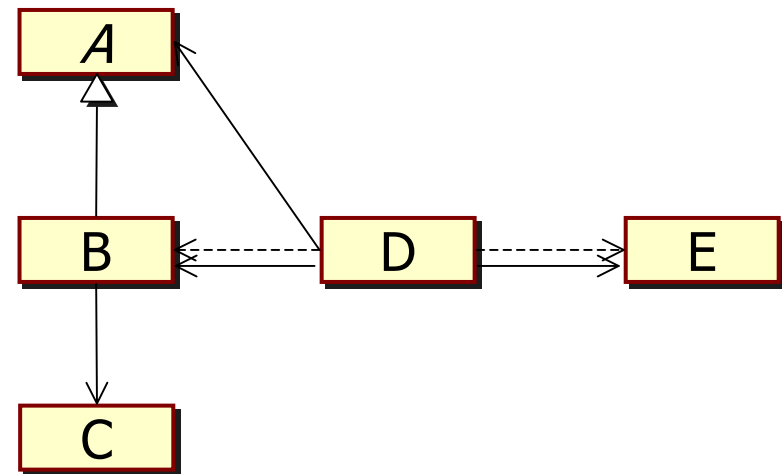
```
public class B extends A {
    public B() { C c; }
}
```

```
public class C {
    private C() {}
}
```

```
public class D {
    public D() {
        A a = new B();
        E e = new E();
    }
}
```

```
public class E {}
```

CRG of example codes



X inherits Y

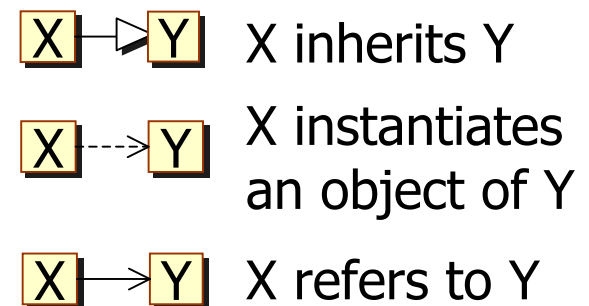
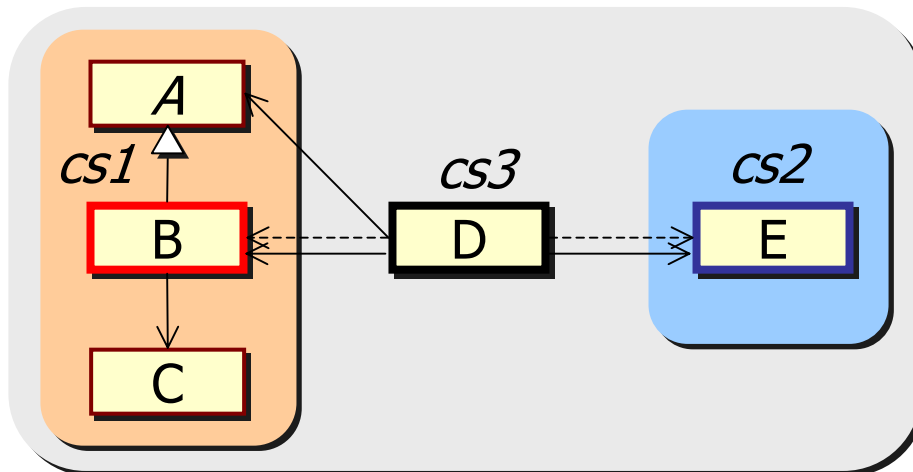
X instantiates an object of Y

X refers to Y

# Detecting Component Candidates

- Structurally reusable component
  - Standalone and executable JavaBeans component
  - Component's interface is separated from implementation
  - All classes necessary for instantiation are packaged
- Detecting all clusters (component candidates) on CRG
  - Cluster:  $cs = ( \langle \text{Facade class} \rangle, \langle \text{Set of participants} \rangle )$

$$cs1 = ( B, \{ A, B, C \} ) \quad cs2 = ( E, \{ E \} ) \quad cs3 = ( D, \{ A, B, C, D, E \} )$$

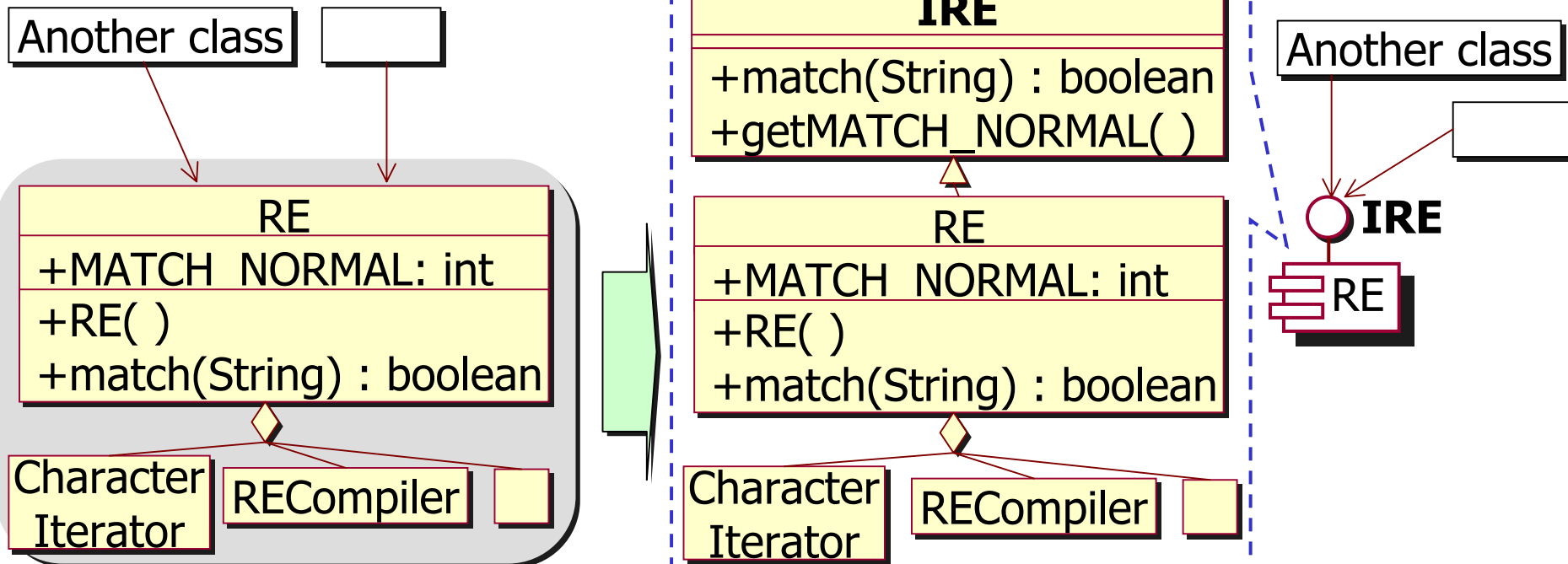


# Extract Component Refactoring

- Transforming Clusters into Components:
  - 1. Creates a new Facade interface
  - 2. Adds declarations of all public methods
  - 3. Adds declarations of setter/getter methods corresponding to public fields
  - 4. Compiles and packages all class files into JAR archive
  - 5. Modifies the surrounding parts to use extracted component

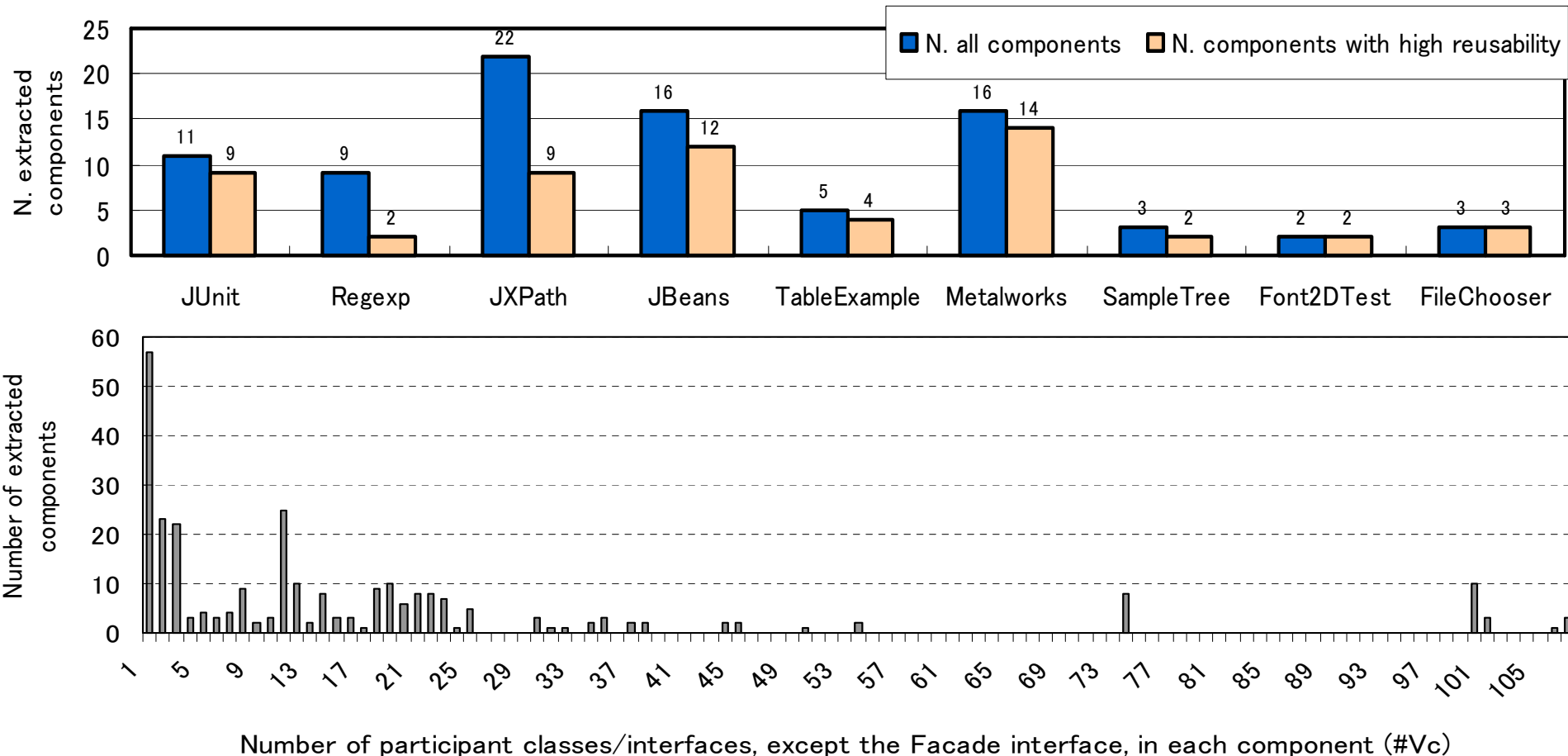
E.g. Jakarta Regexp program

Regular expression component



# Extraction Experiment

- Target: 4 open source programs + 6 Java demo programs
- Result: We extracted 282 components
  - 72% of all extracted components are high reusable
  - 80% of all extracted components are composed of three or more classes



# Automated Extraction Tool

- Implementation: Java2 SDK1.4.0, JavaCC (parser generator)
- Analyzes Java program source codes, and displays CRG
- Performs all necessary steps of the Extract Component Refactoring

The screenshot displays the JExtractor application window. The main area shows a Class Relation Graph (CRG) with the following classes and relationships:

- Class <<test>> AAA**: Contains method `AAA()`. It has an outgoing dependency on **Class <<test>> BBB** labeled `default >>`.
- Class <<test>> BBB**: Contains methods `BBB()`, `BBB(o : Object)`, and `foo(c : CCC) : void`. It has an outgoing dependency on **Class <<test>> CCC** labeled `use >>`.
- Class <<test>> DDD**: Contains methods `DDD(s : String)` and `bar() : BBB`. It has an outgoing dependency on **Class <<test>> BBB** labeled `use >>`.
- Class <<test>> EEE**: Contains method `EEE()`. It has an outgoing dependency on **Class <<test>> AAA** labeled `use >>`.
- Class <<test>> FFF**: Contains method `baz() :`. It has an outgoing dependency on **Class <<test>> CCC** labeled `use >>`.

Below the CRG, the **Dependency Information** section shows:

- Incoming dependency: `test.AAA --> test.BBB (depend)`
- Outgoing dependency: `test.AAA **> test.BBB (default instantiate)`

The **Source code** window shows the following code for `test.BBB.java`:

```
package test;

public class BBB extends CCC {

    public BBB() {
    }
}
```

The **Regions** window shows the following configuration for **Region 1**:

- Facade class name: `test.BBB`
- Classes/Interfaces in Region: `test.BBB`, `test.CCC`, `test.FFF`
- Button: **Extract**

# What does “reusable” mean?

- Extracted components are structurally reusable
  - Internal structure is hidden from outside
  - Component can be instantiated and executed alone
- However, NOT semantically reusable
  - Not always reusable in all possible contexts
  - How to guarantee semantic reusability?
  - Are some generative techniques (e.g. template programming) helpful for this objective?
- Towards “generative reuse”
  - Target of reuse has been abstracted.
  - Target can be easily customized to match the new contexts.
  - Target has capability of generating a part of program by specifying parameters.



# Thank you.

GPCE'04 Software Transformation Systems Workshop  
(Vancouver, Canada, November 24th, 2004)

## Transforming Object-Oriented Programs into Structurally Reusable Components

---

Hironori Washizaki

National Institute of Informatics, Tokyo, Japan

Yoshiaki Fukazawa

Waseda University, Tokyo, Japan

<http://www.washizaki.net/>