

Virtual Modularity and Concern Modeling with FEAT

Martin Robillard
School of Computer Science
McGill University

© Martin P. Robillard 2005

Separation of Concerns Ideal Scenario

Software System

Concern

Modules

© Martin P. Robillard 2005

Separation of Concerns The Sad Reality

Software System

Concern

Modules

© Martin P. Robillard 2005

A Closer Look at the Problem

```

public boolean loadFromFile(String fileName, String backupName) {
    try {
        FileInputStream fis = new FileInputStream(fileName);
        BufferedReader br = new BufferedReader(new InputStreamReader(fis));
        String line = br.readLine();
        while (line != null) {
            // ...
        }
    } catch (IOException e) {
        // Backup recovery code
        // Error-handling code
    }
}

```

IO Code to load a file from disk

Backup recovery code

Error-handling code

© Martin P. Robillard 2005

Why do We Have Bad Modularity Are all software developers inept?

- Limitations of programming languages
 - You can't decompose programs in enough pieces.
- Emergence of unforeseen concerns
 - The customer wants another feature...
- Code decay
 - If you play with it long enough you'll break it.

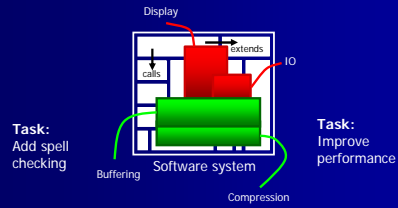
© Martin P. Robillard 2005

Overarching Problem...

Good modularity is a relative concept

© Martin P. Robillard 2005

For Example: A Word Processor



© Martin P. Robillard 2005

7

In the Rest of this Lecture...

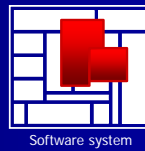
1. Addressing relative modularity
2. Software evolution with FEAT
3. Demo and questions

© Martin P. Robillard 2005

8

Refactoring

Changing the structure of a system without affecting its behavior



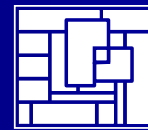
Software system

© Martin P. Robillard 2005

9

Refactoring

Changing the structure of a system without affecting its behavior



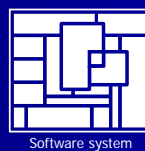
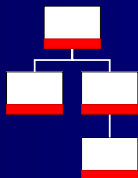
Software system

© Martin P. Robillard 2005

10

Refactoring

Changing the structure of a system without affecting its behavior



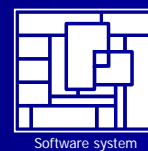
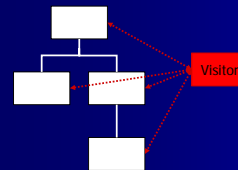
Software system

© Martin P. Robillard 2005

11

Refactoring

Changing the structure of a system without affecting its behavior



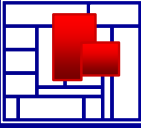
Software system

© Martin P. Robillard 2005

12

Aspect-Oriented Programming

Putting crosscutting structure in modules called aspects



Software system

© Martin P. Robillard 2005 13

Aspect-Oriented Programming

Putting crosscutting structure in modules called aspects

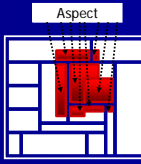
AspectJ Example from a drawing program

```

pointcut move():
call(void FigureElement.setXY(int, int)) ||
call(void Line.setP1(Point)) ||
call(void Line.setP2(Point)) ||
call(void Point.setX(int)) ||
call(void Point.setY(int));

after() returning: move() { dirty = true; }

```



Software system

From *Getting Started with AspectJ*
<http://eclipse.org/aspectj/>

© Martin P. Robillard 2005 14

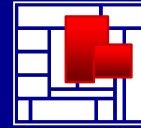
AOP & Refactoring

- Very good when:
 - The “crosscutting” nature of the concern is regular
 - E.g., I’m interested in all calls to methods named “foo”.
- Not so good when:
 - The crosscutting nature of the concern is complex, irregular, fuzzy, etc.
 - I’m sometimes interested in this call to x() in method m() when it may be setting field f...

© Martin P. Robillard 2005 15

Virtual Modularity to the Rescue

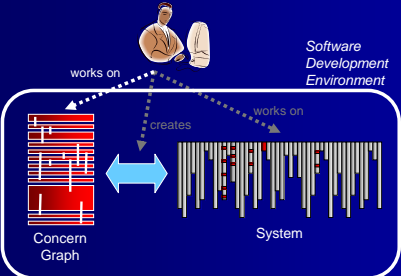
Idea: To use tools to bridge the gap between conceptual and physical modules in a system



Software system

© Martin P. Robillard 2005 16

The Concern Graph Approach



works on

creates

works on

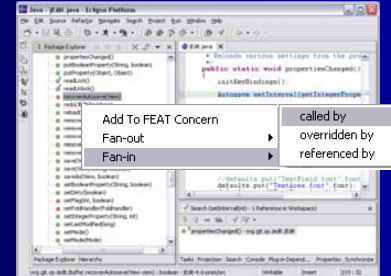
Software Development Environment

Concern Graph

System

© Martin P. Robillard 2005 17

Creating CGs with FEAT



Add To FEAT Concern

Fan-out

Fan-in

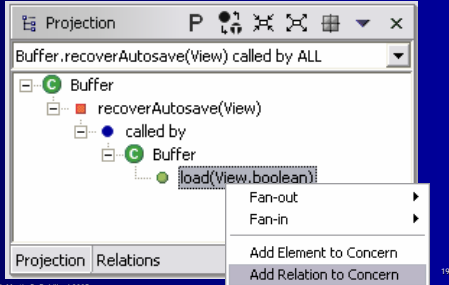
called by

overridden by

referenced by

© Martin P. Robillard 2005 18

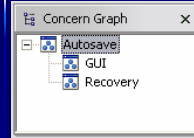
Creating CGs with FEAT



© Martin P. Robillard 2005

19

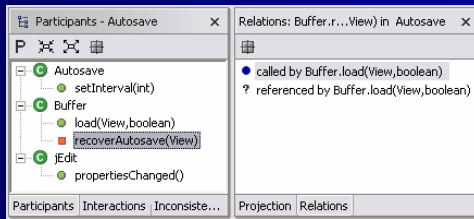
Modifying Systems Using CGs



© Martin P. Robillard 2005

20

Modifying Systems Using CGs



© Martin P. Robillard 2005

21

Modifying Systems Using CGs

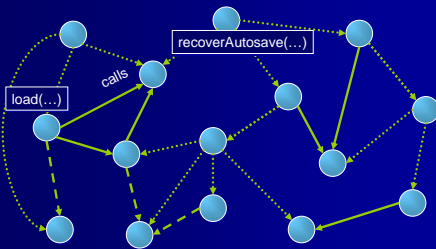
```

Autosave.java
if(reload || !getFlag(REV_FILE))
{
    if(file != null)
        modTime = file.lastModified();
    // Only on initial load
    if(!reload && autosaveFile != null && autosaveFile.exists())
        loadAutosave = recoverAutosave(view);
    else
    {
        if(autosaveFile != null)
            autosaveFile.delete();
        loadAutosave = false;
    }
    if(!loadAutosave)
}
    
```

© Martin P. Robillard 2005

22

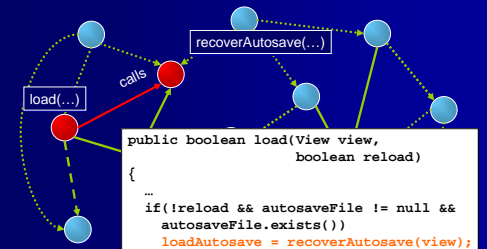
What a Concern Graph Looks Like



© Martin P. Robillard 2005

23

What a Concern Graph Looks Like

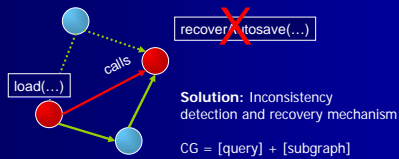


© Martin P. Robillard 2005

24

Maintaining and Reusing CGs

- CGs describe code. What happens to a CG if you change the code?



© Martin P. Robillard 2005

25

Summary

- Badly modularized systems are hard and expensive to change
- Good modularity is relative
- Research problem: to find a cost-effective way to bridge the gap between desired and actual structure.

© Martin P. Robillard 2005

26

Summary

- Potential solution: Virtual modules
 - Models of how high-level concepts map to software artifacts
- Desired Impact: To reduce the complexity of changing software
 - Less effort, fewer bugs

© Martin P. Robillard 2005

27