Virtual Modularity and Concern Modeling with FEAT

Martin Robillard
School of Computer Science
McGill University

Separation of Concerns
Ideal Scenario

Separation of Concerns
The Sad Reality

A Closer Look at the Problem

IO Code to load a file from disk
Backup recovery code
Error-handling code

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.

Overarching Problem...

Good modularity is a relative concept
For Example: A Word Processor

Task: Add spell checking

Task: Improve performance

In the Rest of this Lecture...
1. Addressing relative modularity
2. Software evolution with FEAT
3. Demo and questions

Refactoring
Changing the structure of a system without affecting its behavior

Refactoring
Changing the structure of a system without affecting its behavior

Refactoring
Changing the structure of a system without affecting its behavior

Refactoring
Changing the structure of a system without affecting its behavior
Aspect-Oriented Programming

Putting crosscutting structure in modules called aspects

Software system

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; }

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

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…

Virtual Modularity to the Rescue

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

The Concern Graph Approach

works on

Software Development Environment

works on

Concern Graph

System

Creating CGs with FEAT

Add To FEAT Concern

called by

override by

refers to

For:

© Martin P. Robillard 2005
Creating CGs with FEAT

Modifying Systems Using CGs

What a Concern Graph Looks Like

```java
public boolean load(View view, boolean reload) {
    if(!reload && autosaveFile != null && autosaveFile.exists())
        loadAutosave = recoverAutosave(view);
```
Maintaining and Reusing CGs

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

**Solution:** Inconsistency detection and recovery mechanism

\[ CG = \{query\} + \{subgraph\} \]

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.

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