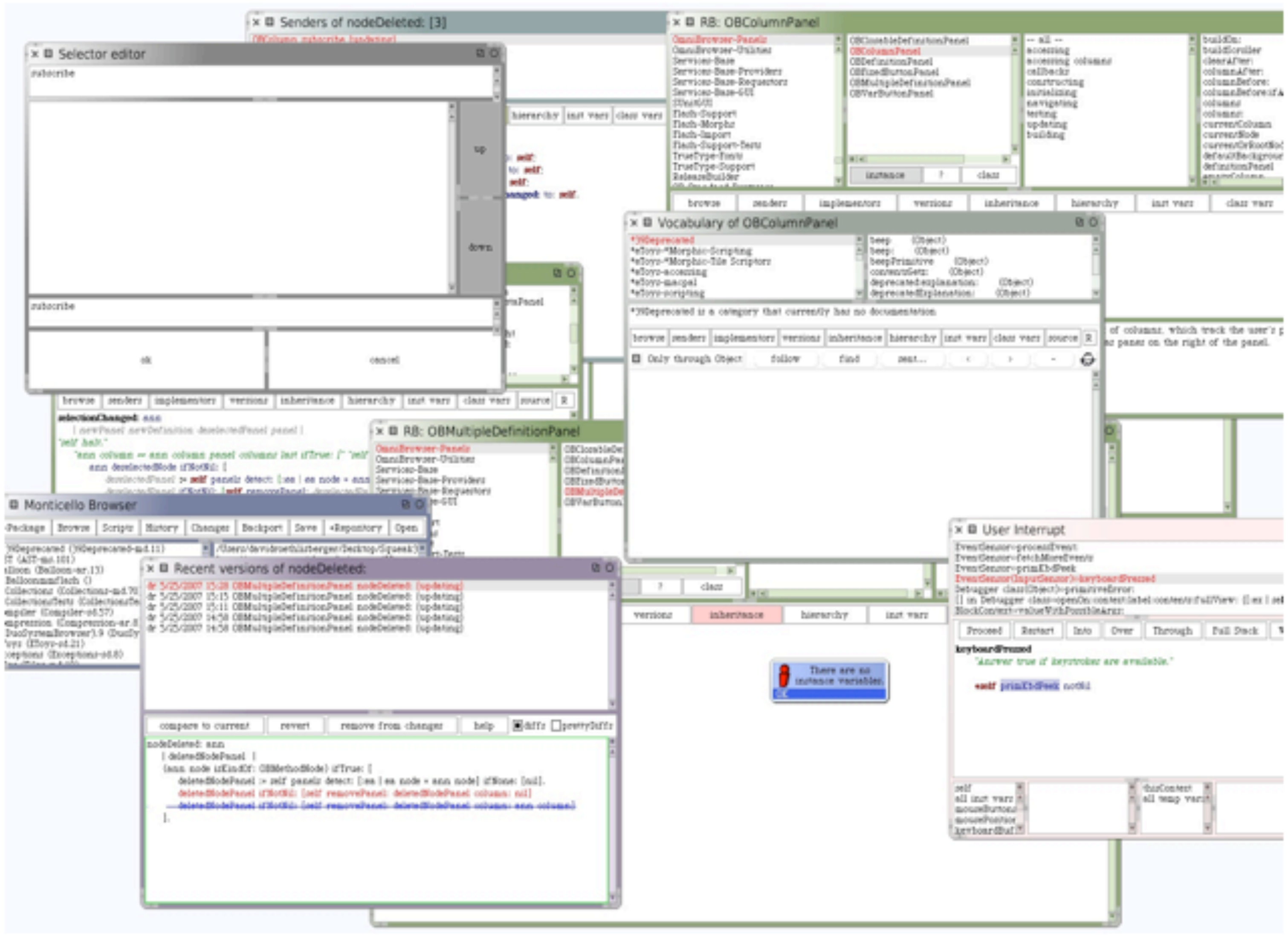
Feature Driven Browsing

David Röthlisberger, Orla Greevy and Oscar Nierstrasz
Software Composition Group, University of Berne

Features

Bugs

Features are not explicit



```
dr 5/25/2007 15:28 OBMultipleDefinitionPanel nodeDeleted: (updating)
dr 5/25/2007 15:15 OBMultipleDefinitionPanel nodeDeleted: (updating)
dr 5/25/2007 15:11 OBMultipleDefinitionPanel nodeDeleted: (updating)
dr 5/25/2007 14:58 OBMultipleDefinitionPanel nodeDeleted: (updating)
dr 5/25/2007 14:58 OBMultipleDefinitionPanel nodeDeleted: (updating)
```

```
nodeDeleted: ann
   | deletedNodePanel |
   (ann node isKindOf: OBMethodNode) ifTrue: [
      deletedNodePanel := self panels detect: [:ea | ea node = ann node] ifNone: [nil].
      deletedNodePanel ifNotNil: [self removePanel: deletedNodePanel column: nil]
      ~~deletedNodePanel ifNotNil: [self removePanel: deletedNodePanel column: ann column]~~
   ].
```

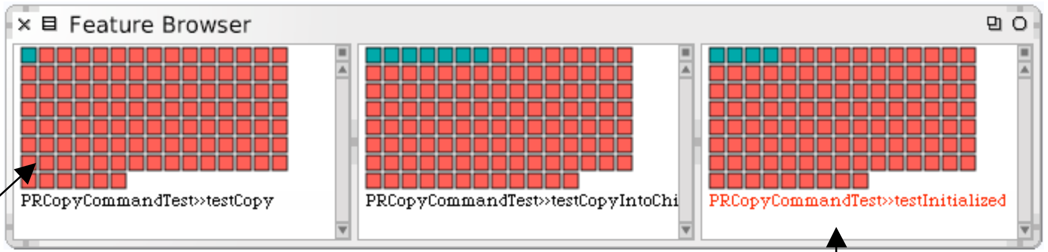
```
keyboardPressed
   "Answer true if keystrokes are available."

   ^self primKbdPeek notNil
```

Solution: Explicit Feature Representation in IDE

- Represent feature as a tree of method invocations
- Nodes are collected dynamically
- Tree is displayed directly in the IDE
- Compare similar features to find abnormalities
- Feature selection:
  - Executing test cases
  - Observing user actions

Feature-Centric Environment

Test Runner
(1)

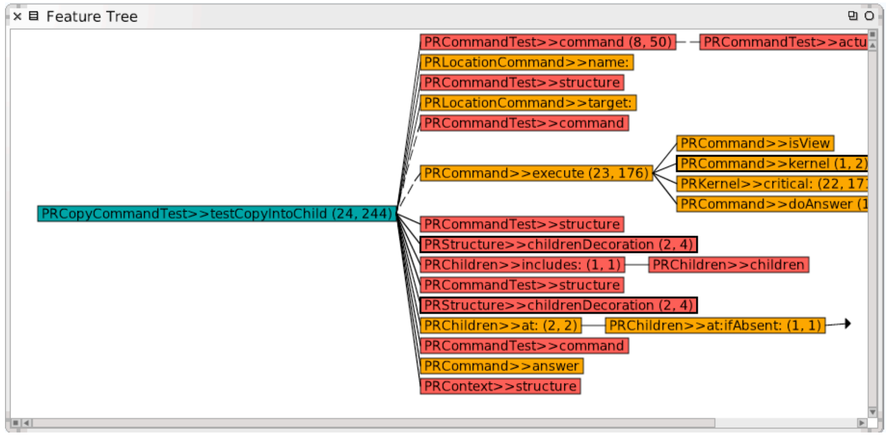


× ⊟ Feature Browser
PRCopyCommandTest>>testCopy
PRCopyCommandTest>>testCopyIntoChi
PRCopyCommandTest>>testInitialized

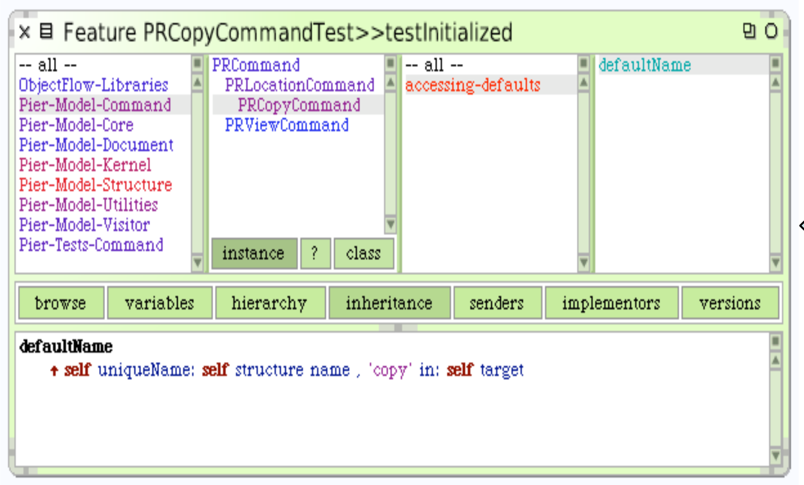
(2) Compact Feature Overview

Invoked method

One feature

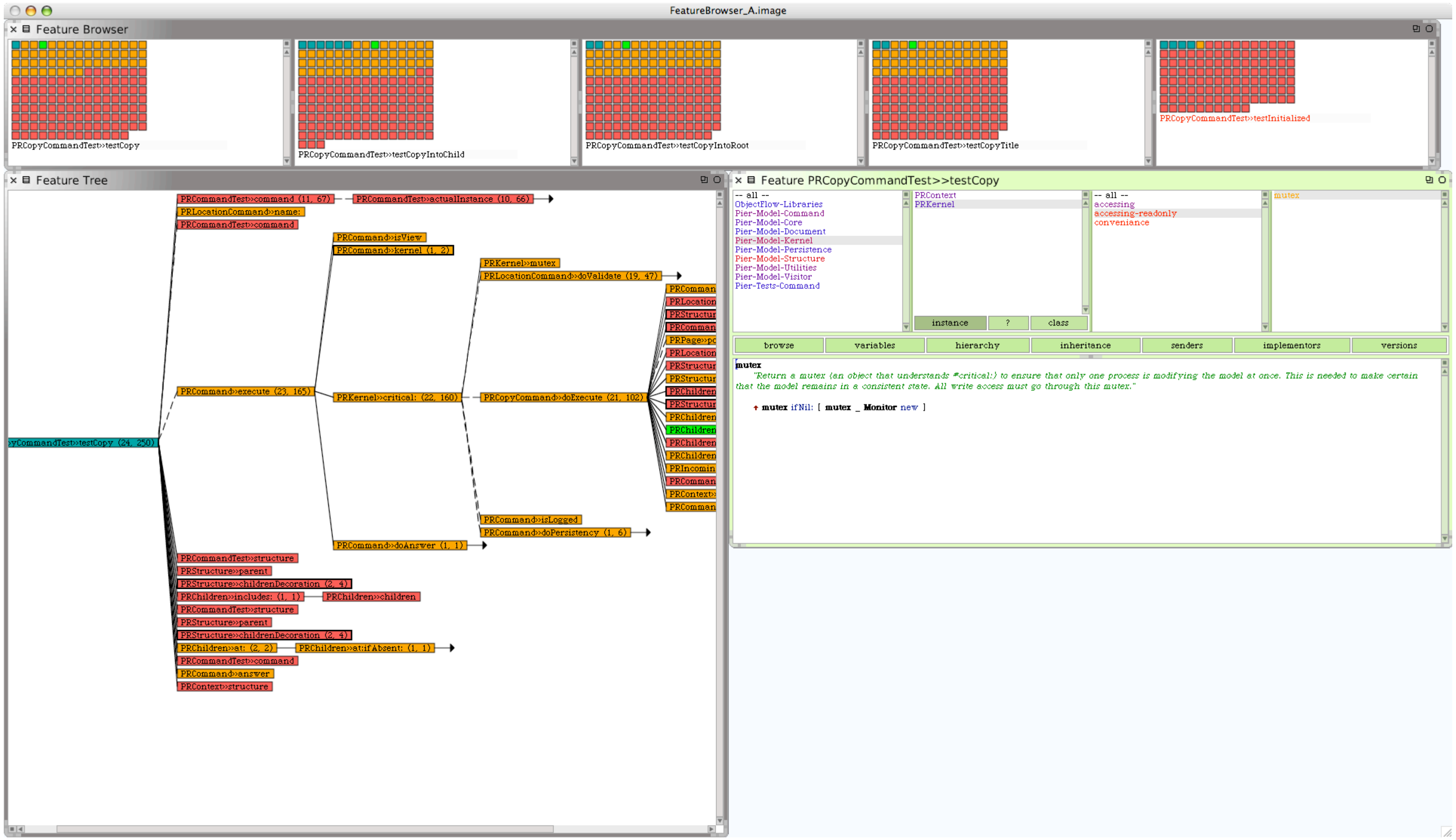


(3) Feature Tree



```
defaultName
	↑ self uniqueName: self structure name , 'copy' in: self target
```

(4) Feature Artifact Browser



```
mutex
	"Return a mutex (an object that understands #critical:) to ensure that only one process is modifying the model at once. This is needed to make certain that the model remains in a consistent state. All write access must go through this mutex."

	↑ mutex ifNil: [ mutex _ Monitor new ]
```

Demo

Empirical Evaluation

- Controlled experiment with 12 students
- Correct two bugs, (i) with traditional Squeak browser, (ii) with Feature-centric Environment.

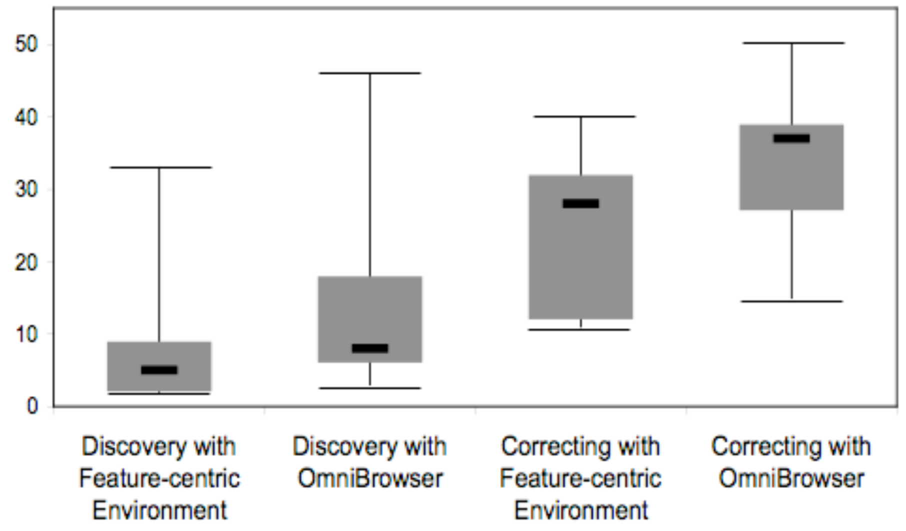
Results:

56% less time spent to find faulty method
33% less time spent to fully correct defect

Threats:

Small number of subjects
Bugs artificially introduced

Results of Experiment



Summary

- Problem:
  - Maintenance of a software system
  - Features not explicitly represented in IDEs
- Solution:
  - Enrich IDEs with explicit feature representation
  - Provide tool support to locate bugs in specific features