

Visualizing and Navigating Source Code History

Alex Bradley

CPSC 533C
University of British Columbia

December 14, 2009

Outline

- 1 Introduction
 - Problem Description
 - Previous Work
- 2 Approach
- 3 Demonstration
- 4 Lessons Learned
- 5 Future Work



Problem Description

- **Domain:** software engineering
- **Task:** compare past revisions of a source code file to learn how it evolved
 - Which authors contributed to a certain section of the code?
 - When was a given bug introduced, and by whom?
 - See “big picture” of code history: when did large changes occur?
 - Major refactorings, introduction of important features. . .
- Two-way comparison currently well-supported, but what about **many-revision** comparisons and visualizing entire code history?

Previous Work: Two-way comparison tools

- Existing interfaces such as Eclipse revision control UI provide very good support for two-way revision comparison

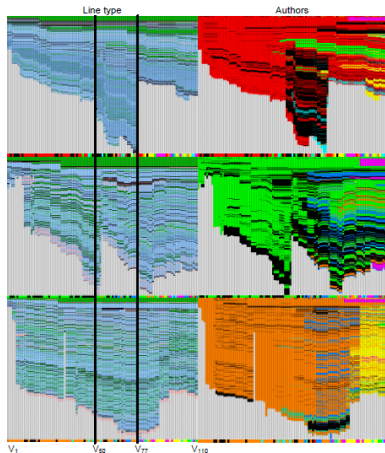
The screenshot displays the Eclipse IDE interface for comparing two versions of the `RefactoringMenuProvider.java` file. The top part shows the `Java Structure Compare` and `Java Source Compare` views. The `Java Source Compare` view shows the code for `RefactoringMenuProvider.java 1.4 [awjb]` on the left and `RefactoringMenuProvider.java 1.3 [awjb]` on the right. The code includes imports, class declarations, and method implementations. The bottom part of the screenshot shows the `Problems` view with a `Revision` table for `RefactoringMenuProvider.java`.

| Revision | Tags | Revision Time | Author | Comment |
|------------|------|-------------------|--------|---|
| Older than | | | | |
| 1.4 | | 08-08-12 5:11 PM | awjb | * For simplicity, phase out top-level refactorings. We should be able to do most of the same stuff with 'contextual' re |
| 1.3 | | 08-08-08 12:19 AM | awjb | Make the menuProvider extension a little less kludgy by providing explicit factory classes. |
| 1.2 | | 08-08-07 10:58 PM | awjb | Now that refactoring stuff is in a separate plugin, add some generics. :-) |
| 1.1 | | 08-08-07 10:16 PM | awjb | Move JQuery-based refactorings out into their own plugin. |

Previous Work: Visual Code Navigator¹, CVSscan²

G. Lommerse¹, F. Nossin¹, L. Voinea^{1,2}, A. Telea^{1,2}, J. J. van Wijk²

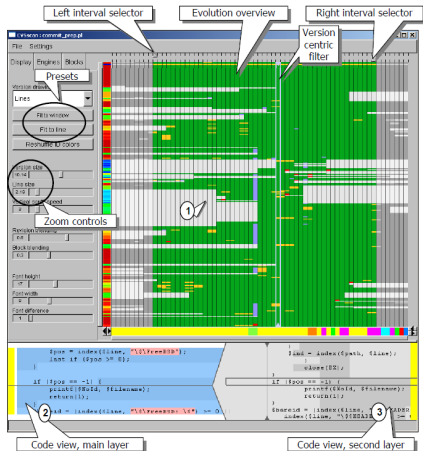
- Flow of vertical revision “stripes”, with lines coloured according to some property of interest
 - Statement type
 - Author of line
 - Differences from other revisions
- Focused on “big picture” visualization; comparison of actual text of revisions less well supported



Previous Work: Visual Code Navigator¹, CVScan²

G. Lommerse¹, F. Nossin¹, L. Voinea^{1,2}, A. Telea^{1,2}, J. J. van Wijk²

- Flow of vertical revision “stripes”, with lines coloured according to some property of interest
 - Statement type
 - Author of line
 - Differences from other revisions
- Focused on “big picture” visualization; comparison of actual text of revisions less well supported

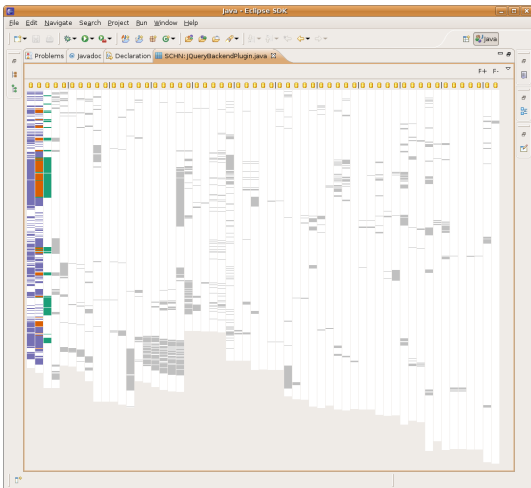


Project Approach

- My solution: “Source Code History Navigator” (SCHN)
- Implemented as Eclipse plugin
- Key techniques:
 - Overview+detail, focus+context
 - Small mid-size multiples
- High-level **overview** of entire revision history provided by “flow” of revision stripes, as in VCN/CVSscan
- Stripes can be “expanded” into text viewers that display **details** of revision code
 - Many text viewers can be opened side-by-side, creating “small multiples” effect

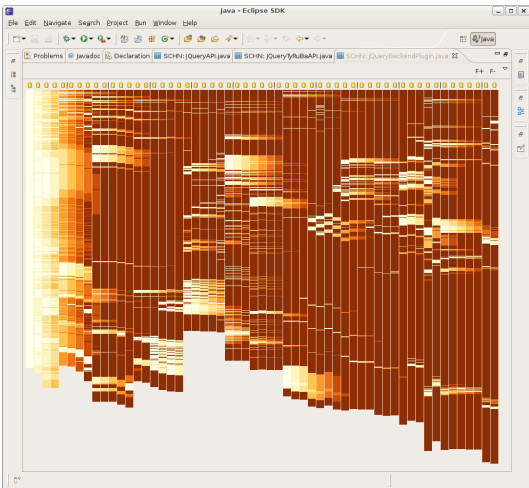
Overview: Revision Flow

57 revisions, lines coloured by differences from neighbouring stripes



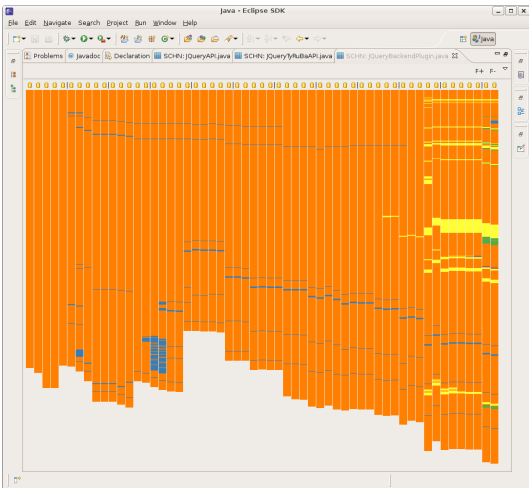
Overview: Revision Flow

57 revisions, lines coloured by code age



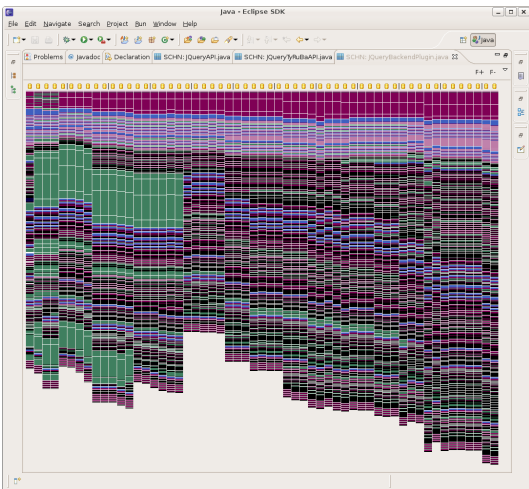
Overview: Revision Flow

57 revisions, lines coloured by authorship



Overview: Revision Flow

57 revisions, lines coloured by statement type



Detail: Many-revision comparison

- Scrollbars of viewers can be locked together for easy navigation
- Text lines coloured (matching stripes) to show differences from neighbours (orange = both ways)

```

/**
 * @see org.osgi.framework.BundleActivator
 */
public void start(BundleContext context) throws Exception {
    super.start(context);

    ISavedState state = ResourcesPlugin.get
    String saveFileName = state.lookup(new
    File f = getStateLocation().append(save
    if (f.exists()) {
        readImportantState(f);
    }

    PLUGIN_VERSION = "*" + context.getBundle
    JQueryBackendPlugin.traceUI("QueryBack

    // get the preferences from the store
    updatePreferences(true);

    // userSelectionListener = new UserSel

    // Set tracing flags
    if (isDebugging()) {
        String option;

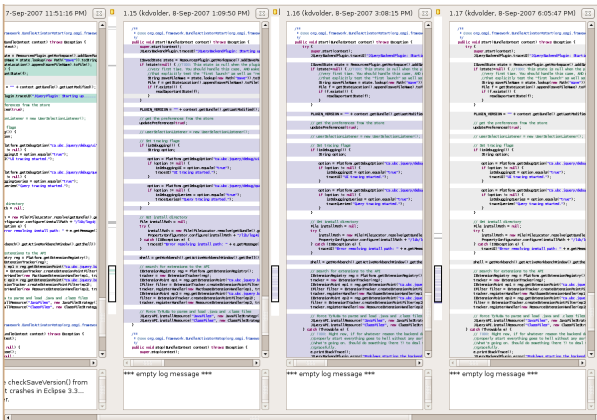
        option = Platform.getDebugOption("
        if (option != null) {
            isDebuggingUI = option.equals("
            traceUI("UI tracing started.");
        }

        option = Platform.getDebugOption("
        if (option != null) {
            isDebuggingQueries = option.equ
            traceQueries("Query tracing sta
        }
    }

    *** empty log message ***
  
```

Mid-level overview

- Reducing font size in text viewers provide “middle ground” between stripe overview and full-size text detail



Demonstration

Lessons: Grid view not very useful

- Did not appear to have much benefit compared to row of revisions
- Comparing many revisions works well with linear visual scan and horizontal scrolling; grid forces pair comparisons and “zigzagging” scan

```

1.4 [web, 12-Aug-2008 5:11:59 PM]
public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "Go.Ubc.Jquery.JQueryTreeview...
    private static final String REFACTORING_QUERY = "refactoringTargets, fLabels,
    private static final String REFACTORING_QUERY_ORIGINAL = "fLabels";
    public RefactoringMenuProvider (JQueryTreeViewer view,
        IDMenuManager menuContext) {
        super(view, menuContext);
    }

    @Override
    public void addMenuGroup(IDMenuManager menu) {
        menu.addMenuSeparator(GROUP_REFACTORING);
    }

    @Override
    public void addVisibleMenu(IDMenuManager menu,
        IStructureSelection selection) {
        ITarget[] targets = findTargetQueryNodes(selection);
        if (targets.length == 0) {
            IDMenuManager refactorMenu = new MenuManager("Refactor");
            addAvailableRefactoring(refactoringMenu, targets);
            menu.append(refactorMenu, ITarget[]) {}
        }
    }
}

1.3 [web, 8-Aug-2008 12:19:13 AM]
public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "Go.Ubc.Jquery.JQueryTreeview...
    public RefactoringMenuProvider (JQueryTreeViewer view,
        IDMenuManager menuContext) {
        super(view, menuContext);
    }

    @Override
    public void addVisibleMenu(IDMenuManager menu,
        IStructureSelection selection) {
        addContextualRefactoring(menu);
        addContextualRefactoring(menu, selection);
    }

    @Override
    public void addMenuGroup(IDMenuManager menu) {
        menu.addMenuSeparator(GROUP_REFACTORING);
    }

    private void addTopLevelRefactorings (IDMenuManager menu) {
        // can't do top queries if the first isn't rooted at the factories
        if (view.isRootTree()) {
            IDMenuManager topRefactorings = new MenuManager("Available Top-level Re
            menu.appendToGroup(GROUP_REFACTORING, topRefactorings);
        }
    }
}

1.2 [web, 7-Aug-2008 10:58:25 PM]
public class RefactoringMenuProvider extends ExtensionMenuProvider {
    public static final String GROUP_REFACTORING = "Go.Ubc.Jquery.JQueryTreeview...

    public RefactoringMenuProvider () {
        super();
    }

    public RefactoringMenuProvider (JQueryTreeViewer view,
        IDMenuManager menuContext) {
        super(view, menuContext);
    }

    @Override
    public ExtensionMenuProvider getInstance (JQueryTreeViewer view,
        IDMenuManager menuContext) {
        return new RefactoringMenuProvider(view, menuContext);
    }

    @Override
    public void addVisibleMenu(IDMenuManager menu,
        IStructureSelection selection) {}
}

1.1 [web, 7-Aug-2008 10:16:09 PM]
public class RefactoringMenuProvider extends ExtensionMenuProvider {
    public static final String GROUP_REFACTORING = "Go.Ubc.Jquery.JQueryTreeview...
    public RefactoringMenuProvider () {
        super();
    }

    public RefactoringMenuProvider (JQueryTreeViewer view,
        IDMenuManager menuContext) {
        super(view, menuContext);
    }

    @Override
    public ExtensionMenuProvider getInstance (JQueryTreeViewer view,
        IDMenuManager menuContext) {
        return new RefactoringMenuProvider(view, menuContext);
    }

    @Override
    public void addVisibleMenu(IDMenuManager menu,
        IStructureSelection selection) {}
}

```

* For simplicity, phase out top-level refactorings. We should be able to do most of the same stuff with "contextual" refactorings, and the latter are closer in style to Eclipse's existing refactoring menus.

* JQueryRefactoringTypeBuilder type no longer used - its fields are merged into the single "refactoring" attribute via mixins.

* Make the menuProvider extension a little less kludgy by providing explicit factory classes.

Lessons: Grid view not very useful

- Did not appear to have much benefit compared to row of revisions
- Comparing many revisions works well with linear visual scan and horizontal scrolling; grid forces pair comparisons and “zigzagging” scan

The screenshot shows an IDE window titled "Multi-Revision View" displaying a grid of source code revisions for the file "RefactoringMenuProvider.java". The grid is organized into columns for different revisions: 1.4 (Aug-2008 5:11:59 PM), 1.3 (Aug-2008 12:19:13 AM), and 1.2 (Aug-2008 10:58:25 PM). Each column contains the code for that revision. A large red 'X' is drawn over the grid, signifying that this grid view is not useful for comparing many revisions. Below the code, there are some comments and a small table of revisions.

```

1.4 [auh, 12-Aug-2008 5:11:59 PM]
public class RefactoringMenuProvider extends AbstractLayeredMenuProvider {
    public static final String GROUP_REFACTORING = "org.eclipse.jdt.ui.RefactoringMenu";
    private static final String ID_REFACTORING = "org.eclipse.jdt.ui.RefactoringMenu";
    public RefactoringMenuProvider (JQueryTreeViewer view, IMenuManager menuContext) {
        super(view, menuContext);
    }
    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }
    @Override
    public void addVisibleItem(IMenuManager menu,
        IStructureSelection selection) {
        ITarget targets = selection.getTarget();
        if (targets.length == 0) {
            IMenuManager refactoringsMenu = new MenuManager("Refactorings");
            menu.add(new RefactoringMenu(refactoringsMenu, targets));
        }
    }
}

1.3 [auh, 8-Aug-2008 12:19:13 AM]
public class RefactoringMenuProvider extends AbstractLayeredMenuProvider {
    public static final String GROUP_REFACTORING = "org.eclipse.jdt.ui.RefactoringMenu";
    public RefactoringMenuProvider (JQueryTreeViewer view, IMenuManager menu,
        IMenuManager menuContext) {
        super(view, menu, menuContext);
    }
    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }
    @Override
    public void addVisibleItem(IMenuManager menu,
        IStructureSelection selection) {
        ITarget targets = selection.getTarget();
        if (targets.length == 0) {
            IMenuManager refactoringsMenu = new MenuManager("Refactorings");
            menu.add(new RefactoringMenu(refactoringsMenu, targets));
        }
    }
}

1.2 [auh, 7-Aug-2008 10:58:25 PM]
public class RefactoringMenuProvider extends AbstractLayeredMenuProvider {
    public static final String GROUP_REFACTORING = "org.eclipse.jdt.ui.RefactoringMenu";
    public RefactoringMenuProvider () {
        super();
    }
    public RefactoringMenuProvider (JQueryTreeViewer view, IMenuManager menuContext) {
        super(view, menuContext);
    }
    @Override
    public IMenuItem getMenuItem(IMenuManager menu,
        IStructureSelection selection) {
        return new RefactoringMenuProvider(view, menuContext);
    }
    @Override
    public void addVisibleItem(IMenuManager menu,
        IStructureSelection selection) {
    }
}

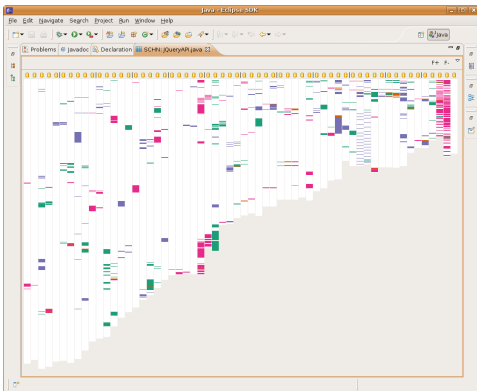
```

For simplicity, phase out top-level refactorings. We should be able to do the same stuff with "contextual" refactorings, and the latter are closer to existing refactoring menus.
 * "QueryRefactoring" type is no longer used - its fields are unused.
 * "Refactoring" predicate see now use.

| Revision | Author | Date |
|----------|--------|------------------------|
| 1.4 | auh | 12-Aug-2008 5:11:59 PM |
| 1.3 | auh | 8-Aug-2008 12:19:13 AM |
| 1.2 | auh | 7-Aug-2008 10:58:25 PM |

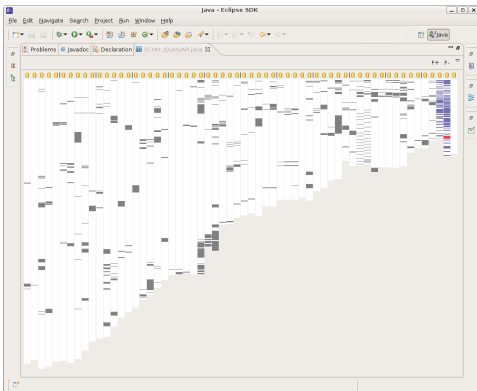
Lessons: Colours are tricky

- Large patches of saturated colour in author-of-line view grab attention and make it difficult to spot small changes
 - One-pixel lines can be tricky to spot in general
- Multiple attempts at colouring neighbour differences



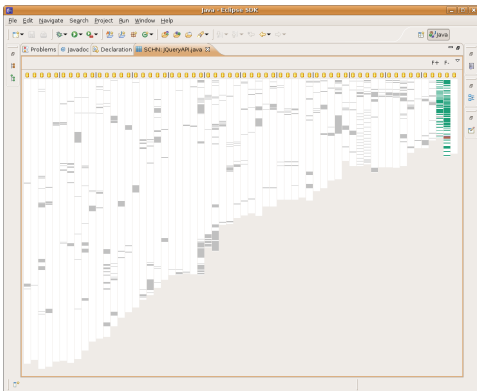
Lessons: Colours are tricky

- Large patches of saturated colour in author-of-line view grab attention and make it difficult to spot small changes
 - One-pixel lines can be tricky to spot in general
- Multiple attempts at colouring neighbour differences



Lessons: Colours are tricky

- Large patches of saturated colour in author-of-line view grab attention and make it difficult to spot small changes
 - One-pixel lines can be tricky to spot in general
- Multiple attempts at colouring neighbour differences



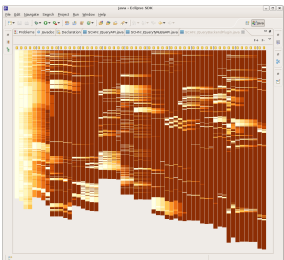
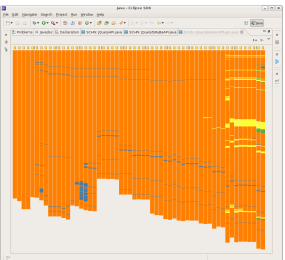
Future Work

- Improvements to **visualization techniques**
 - Keep improving presentation of neighbour differences
 - Mouseover highlights for individual difference pairs
 - Character-level differences in text viewers (currently only line diffs)
 - How to present deletions?
 - Add legends for colours used (e.g. colour-to-author mapping)
 - Scaling of stripes
 - Scale down stripes too big for window (how?)
 - Scale up a flow of short stripes to fit window
- Improve **performance**
 - Implement caching of CVS data (and computed diffs?)
 - Optimize graphical rendering
- **Validation:** Apply to more real codebases, do user studies

References/Image Sources

- G. Lommerse, F. Nossin, L. Voinea, and A. Telea. The visual code navigator: An interactive toolset for source code investigation. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 4, Washington, DC, USA, 2005. IEEE Computer Society.
- L. Voinea, A. Telea, and J. J. van Wijk. CVSscan: visualization of code evolution. In *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*, pages 47–56, New York, NY, USA, 2005. ACM.

Questions?



```
1 public void start(@Context context) {  
2     super.start(context);  
3  
4     //initialize state in ResourceRegistry  
5     String nameFlavour = state.lookupNew  
6     <String> f = getReferenceContext().appendName  
7     readResourceState();  
8  
9     flushMessages(); // context.getMsgs()  
10    registerResourceRegistryContextResource();  
11  
12    // get the preferences from the store  
13    updateReferenceContext();  
14  
15    // initialize the customer = new Object();  
16  
17    // Set tracing flag  
18    if (isDebugEnabled()) {  
19        String options = Platform.getProperty()  
20        if (isOption("melli") &  
21            !isDebugEnabled()) {  
22            LogManager = option.equals("melli") ?  
23                LogManager.getInstance("melli") :  
24                LogManager.getInstance("org.eclipse.jdt.core") ;  
25        }  
26        LogManager.addAppender(new ConsoleAppender());  
27        LogManager.setLevel(LogManager.getLevelForName("TRACE"));  
28        LogManager.start("Tracing started.");  
29    }  
30  
31    option = Platform.getProperty()  
32    if (isOption("melli") &  
33        !isDebugEnabled()) {  
34        LogManager.addAppender = option.equals("melli") ?  
35            LogManager.getInstance("melli") :  
36            LogManager.getInstance("org.eclipse.jdt.core") ;  
37    }  
38  
39    // Log files  
40    // to enable checkAccessControl from "Subst" when it  
41    // does it output 23... will be this later.  
42  
43    // empty log message  
44
```

```
1 public void start(@Context context) {  
2     super.start(context);  
3  
4     //initialize state in ResourceRegistry  
5     String nameFlavour = state.lookupNew  
6     <String> f = getReferenceContext().appendName  
7     readResourceState();  
8  
9     flushMessages(); // context.getMsgs()  
10    registerResourceRegistryContextResource();  
11  
12    // get the preferences from the store  
13    updateReferenceContext();  
14  
15    // initialize the customer = new Object();  
16  
17    // Set tracing flag  
18    if (isDebugEnabled()) {  
19        String options = Platform.getProperty()  
20        if (isOption("melli") &  
21            !isDebugEnabled()) {  
22            LogManager = option.equals("melli") ?  
23                LogManager.getInstance("melli") :  
24                LogManager.getInstance("org.eclipse.jdt.core") ;  
25        }  
26        LogManager.addAppender(new ConsoleAppender());  
27        LogManager.setLevel(LogManager.getLevelForName("TRACE"));  
28        LogManager.start("Tracing started.");  
29    }  
30  
31    option = Platform.getProperty()  
32    if (isOption("melli") &  
33        !isDebugEnabled()) {  
34        LogManager.addAppender = option.equals("melli") ?  
35            LogManager.getInstance("melli") :  
36            LogManager.getInstance("org.eclipse.jdt.core") ;  
37    }  
38  
39    // Log files  
40    // to enable checkAccessControl from "Subst" when it  
41    // does it output 23... will be this later.  
42  
43    // empty log message  
44
```