Visualizing Source Code History

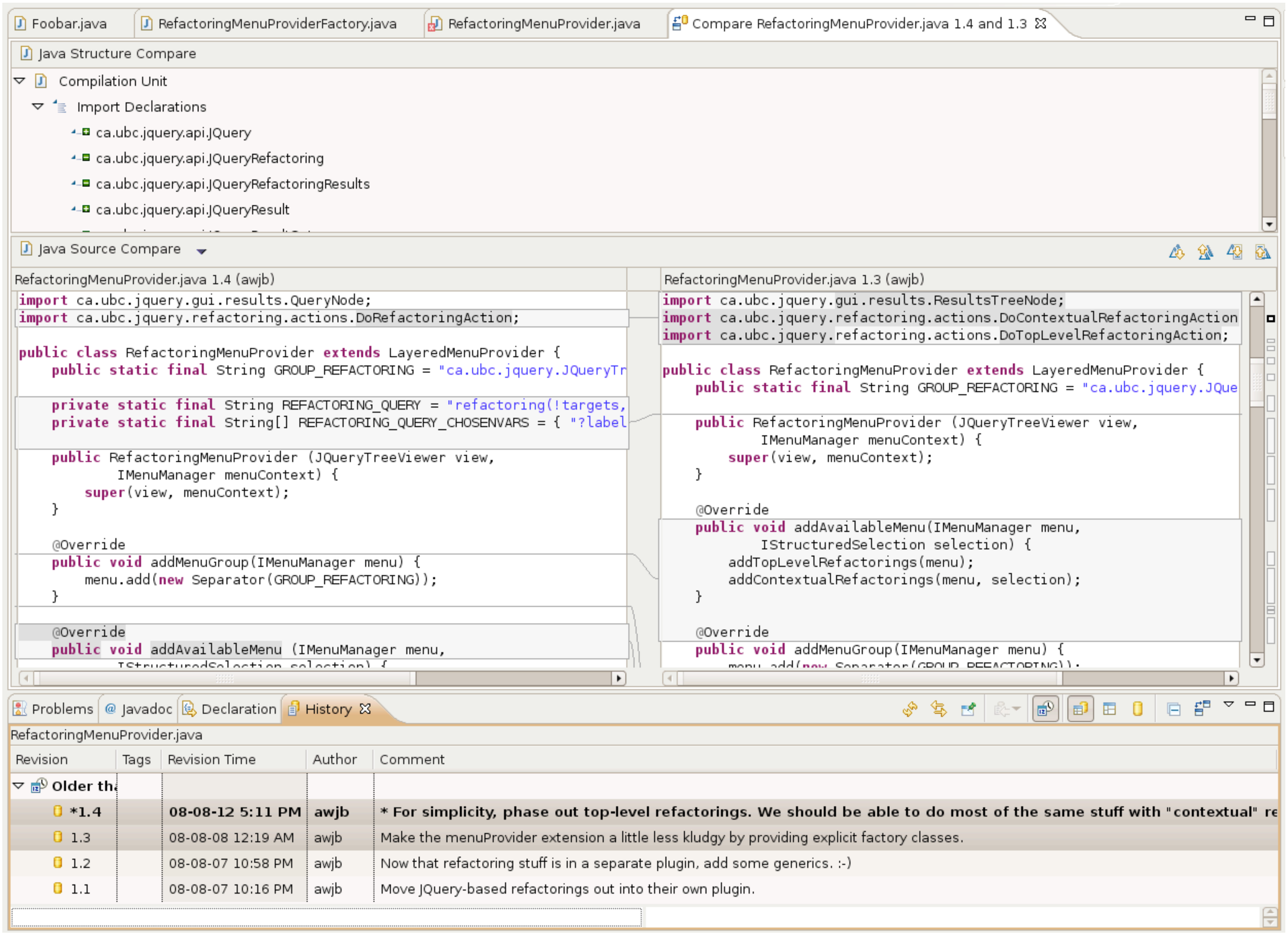
Alex Bradley

CPSC 533C Project Update
University of British Columbia
November 18, 2009

Problem

- Common software development task: compare revisions of a source code file to identify important changes
  - See how a feature evolved over time
  - See which authors participated in development of the code
  - Find when a bug was introduced (and by whom)

Problem



Foobar.java | RefactoringMenuProviderFactory.java | RefactoringMenuProvider.java | Compare RefactoringMenuProvider.java 1.4 and 1.3

Java Structure Compare

- Compilation Unit
  - Import Declarations
    - ca.ubc.jquery.api.JQuery
    - ca.ubc.jquery.api.JQueryRefactoring
    - ca.ubc.jquery.api.JQueryRefactoringResults
    - ca.ubc.jquery.api.JQueryResult

Java Source Compare

RefactoringMenuProvider.java 1.4 (awjb)

```
import ca.ubc.jquery.gui.results.QueryNode;
import ca.ubc.jquery.refactoring.actions.DoRefactoringAction;

public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTr

    private static final String REFACTORING_QUERY = "refactoring(!targets,
    private static final String[] REFACTORING_QUERY_CHOSENVARS = { "?label

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

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    @Override
    public void addAvailableMenu (IMenuManager menu,
            IStructuredSelection selection) {
```

RefactoringMenuProvider.java 1.3 (awjb)

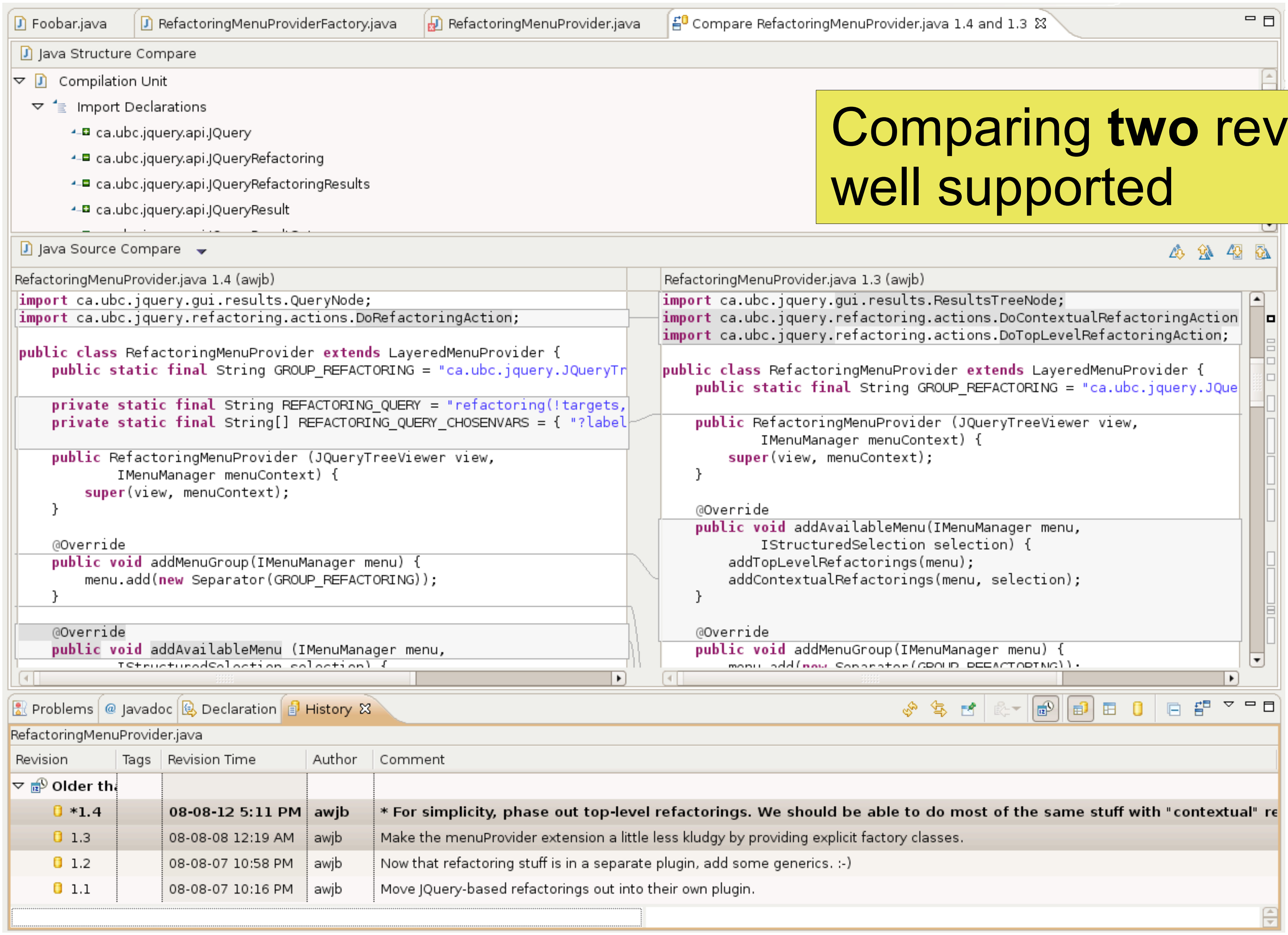
```
import ca.ubc.jquery.gui.results.ResultsTreeNode;
import ca.ubc.jquery.refactoring.actions.DoContextualRefactoringAction
import ca.ubc.jquery.refactoring.actions.DoTopLevelRefactoringAction;

public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQue

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
```

Problems | Javadoc | Declaration | History

RefactoringMenuProvider.java

Revision	Tags	Revision Time	Author	Comment
Older th				
***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.

Problem



Foobar.java | RefactoringMenuProviderFactory.java | RefactoringMenuProvider.java | Compare RefactoringMenuProvider.java 1.4 and 1.3

Java Structure Compare

- Compilation Unit
  - Import Declarations
    - ca.ubc.jquery.api.JQuery
    - ca.ubc.jquery.api.JQueryRefactoring
    - ca.ubc.jquery.api.JQueryRefactoringResults
    - ca.ubc.jquery.api.JQueryResult

Java Source Compare

RefactoringMenuProvider.java 1.4 (awjb)

```
import ca.ubc.jquery.gui.results.QueryNode;
import ca.ubc.jquery.refactoring.actions.DoRefactoringAction;

public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTr

    private static final String REFACTORING_QUERY = "refactoring(!targets,
    private static final String[] REFACTORING_QUERY_CHOSENVARS = { "?label

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

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    @Override
    public void addAvailableMenu (IMenuManager menu,
            IStructuredSelection selection) {
```

RefactoringMenuProvider.java 1.3 (awjb)

```
import ca.ubc.jquery.gui.results.ResultsTreeNode;
import ca.ubc.jquery.refactoring.actions.DoContextualRefactoringAction
import ca.ubc.jquery.refactoring.actions.DoTopLevelRefactoringAction;

public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQue

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
```

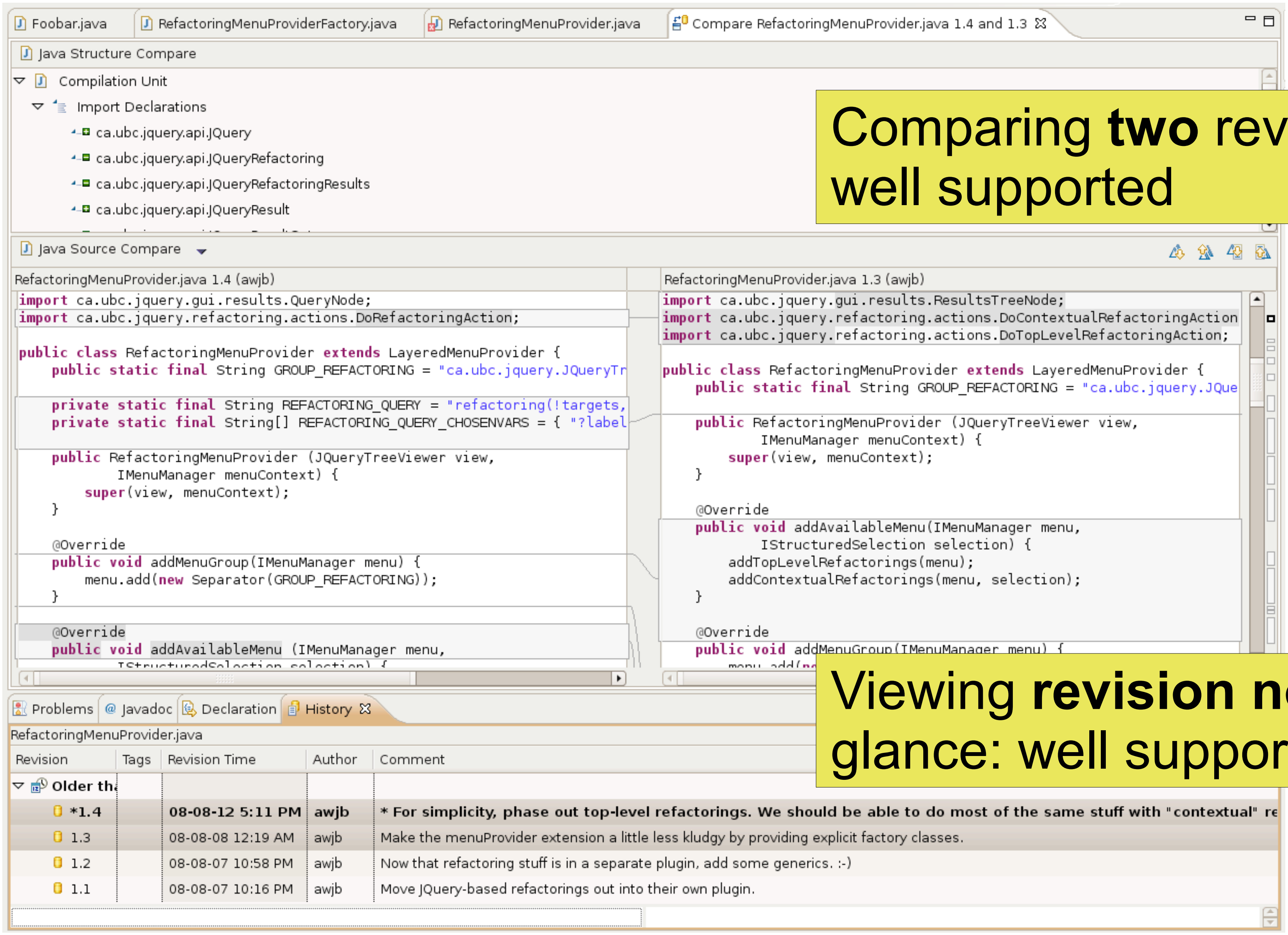
Problems | Javadoc | Declaration | History

RefactoringMenuProvider.java

Revision	Tags	Revision Time	Author	Comment
Older th				
***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.

Comparing **two** revisions: well supported

Problem



Foobar.java | RefactoringMenuProviderFactory.java | RefactoringMenuProvider.java | Compare RefactoringMenuProvider.java 1.4 and 1.3

Java Structure Compare

```
Compilation Unit
  Import Declarations
    ca.ubc.jquery.api.JQuery
    ca.ubc.jquery.api.JQueryRefactoring
    ca.ubc.jquery.api.JQueryRefactoringResults
    ca.ubc.jquery.api.JQueryResult
```

Java Source Compare

RefactoringMenuProvider.java 1.4 (awjb)

```
import ca.ubc.jquery.gui.results.QueryNode;
import ca.ubc.jquery.refactoring.actions.DoRefactoringAction;

public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTr

    private static final String REFACTORING_QUERY = "refactoring(!targets,
    private static final String[] REFACTORING_QUERY_CHOSENVARS = { "?label

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

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    @Override
    public void addAvailableMenu (IMenuManager menu,
```

RefactoringMenuProvider.java 1.3 (awjb)

```
import ca.ubc.jquery.gui.results.ResultsTreeNode;
import ca.ubc.jquery.refactoring.actions.DoContextualRefactoringAction
import ca.ubc.jquery.refactoring.actions.DoTopLevelRefactoringAction;

public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQue

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
```

Problems | Javadoc | Declaration | History

RefactoringMenuProvider.java

Revision	Tags	Revision Time	Author	Comment
Older th				
*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.

Comparing **two** revisions: well supported

Viewing **revision notes** at a glance: well supported

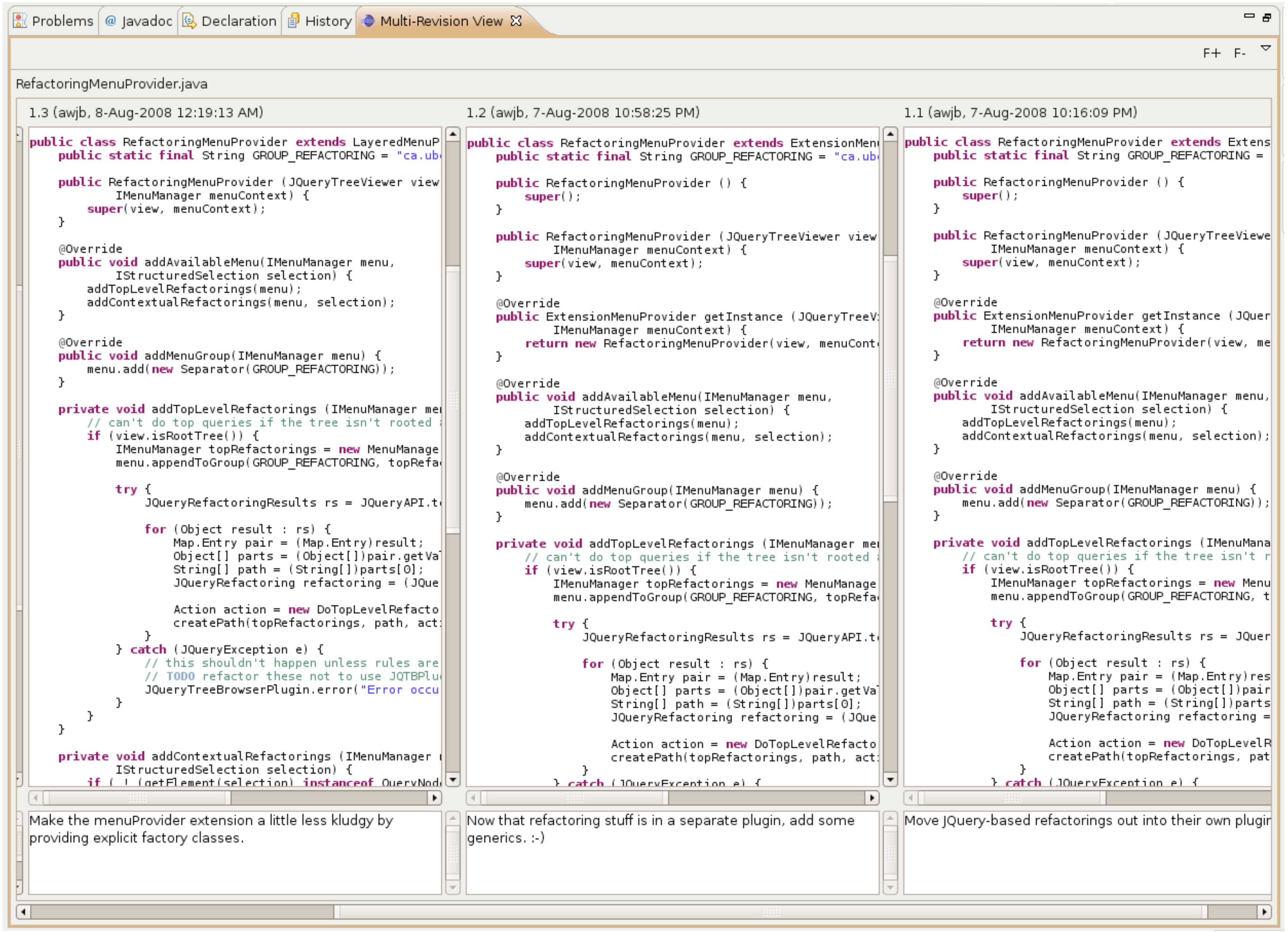
Problem

- But how to compare **many** revisions?
  - Look for evolution of code over several revisions of concern (in my experience, ~4-10)
  - Look at overall picture of all revisions to find where interesting stuff might have happened

Solution: Basic Ideas

- Small multiples view for detailed comparison
  - Two alternatives considered:
    - Revisions side by side in a row
    - Revisions in a two-column grid
- “History flow” to display all revisions
  - Each revision as a vertical pixel stripe
  - Lines in revision = horizontal pixel lines in stripe
    - Coloured according to authorship, statement type, code age...
  - Full-text views of a few revisions of interest embedded in flow (focus+context)

Prototype: single row view



Problems | Javadoc | Declaration | History | Multi-Revision View

F+ F-

RefactoringMenuProvider.java

1.3 (awjb, 8-Aug-2008 12:19:13 AM)

```
public class RefactoringMenuProvider extends LayeredMenuP
    public static final String GROUP_REFACTORING = "ca.ub

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    private void addTopLevelRefactorings (IMenuManager men
        // can't do top queries if the tree isn't rooted
        if (view.isRootTree()) {
            IMenuManager topRefactorings = new MenuManage
            menu.appendToGroup(GROUP_REFACTORING, topRefa

            try {
                JQueryRefactoringResults rs = JQueryAPI.t

                for (Object result : rs) {
                    Map.Entry pair = (Map.Entry)result;
                    Object[] parts = (Object[])pair.getVa
                    String[] path = (String[])parts[0];
                    JQueryRefactoring refactoring = (JQue

                    Action action = new DoTopLevelRefacto
                    createPath(topRefactorings, path, act
                }
            } catch (JQueryException e) {
                // this shouldn't happen unless rules are
                // TODO refactor these not to use JQTBPlu
                JQueryTreeBrowserPlugin.error("Error occu
            }
        }
    }

    private void addContextualRefactorings (IMenuManager
            IStructuredSelection selection) {
        if ( ! (getElement(selection) instanceof QueryNod
```

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

1.2 (awjb, 7-Aug-2008 10:58:25 PM)

```
public class RefactoringMenuProvider extends ExtensionMen
    public static final String GROUP_REFACTORING = "ca.ub

    public RefactoringMenuProvider () {
        super();
    }

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

    @Override
    public ExtensionMenuProvider getInstance (JQueryTreeV
            IMenuManager menuContext) {
        return new RefactoringMenuProvider(view, menuCont
    }

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    private void addTopLevelRefactorings (IMenuManager me
        // can't do top queries if the tree isn't rooted
        if (view.isRootTree()) {
            IMenuManager topRefactorings = new MenuManage
            menu.appendToGroup(GROUP_REFACTORING, topRefa

            try {
                JQueryRefactoringResults rs = JQueryAPI.t

                for (Object result : rs) {
                    Map.Entry pair = (Map.Entry)result;
                    Object[] parts = (Object[])pair.getVa
                    String[] path = (String[])parts[0];
                    JQueryRefactoring refactoring = (JQue

                    Action action = new DoTopLevelRefacto
                    createPath(topRefactorings, path, act
                }
            } catch (JQueryException e) {
```

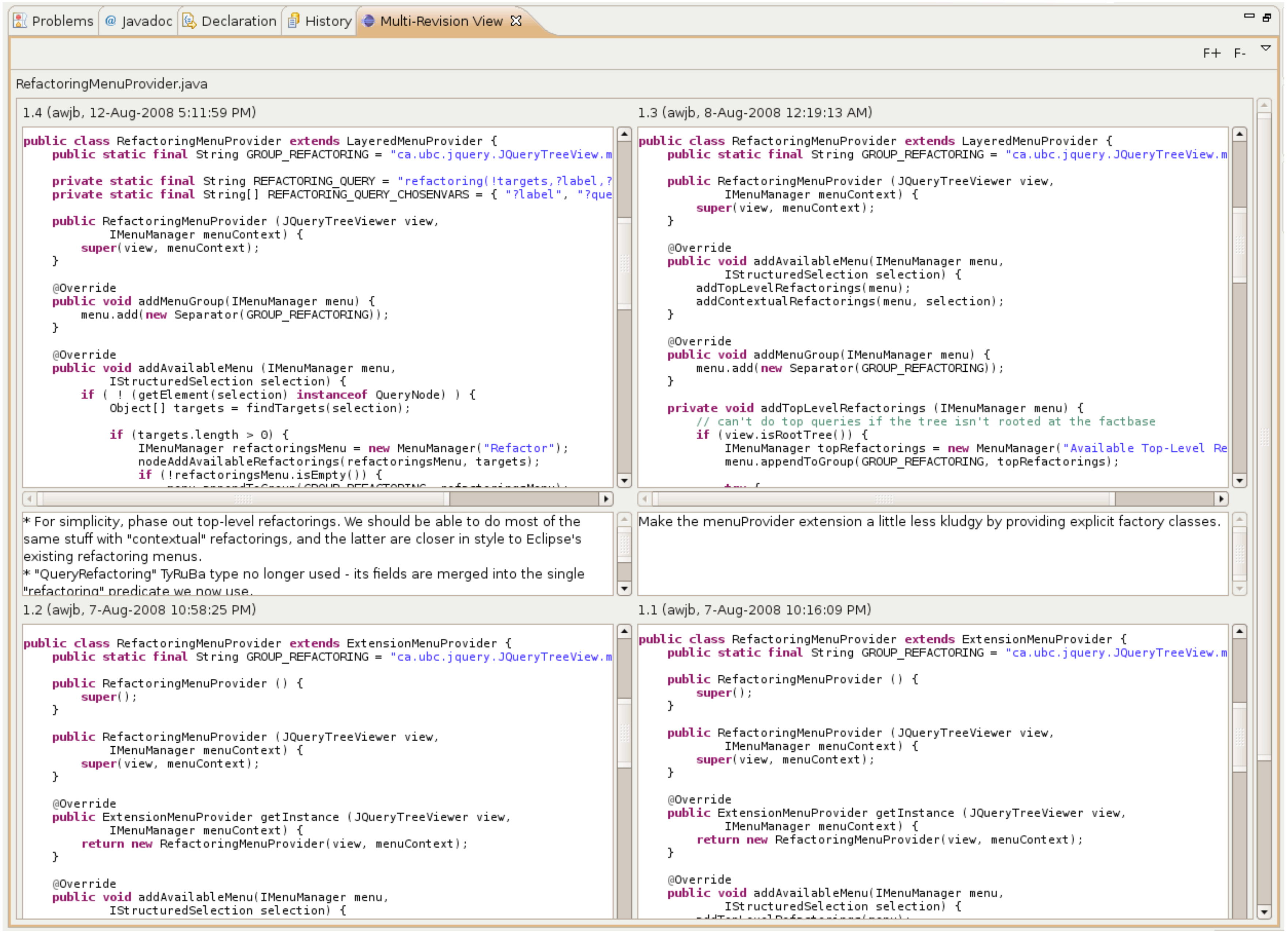
Now that refactoring stuff is in a separate plugin, add some generics. :-)

1.1 (awjb, 7-Aug-2008 10:16:09 PM)

```
public class RefactoringMenuProvider extends Extens
    public static final String GROUP_REFACTORING =

    public RefactoringMenuProvider () {
        super();
    }

    public RefactoringMenuProvider (JQueryTreeViewe
            IMenuManager menuContext) {
        super(view, menuContext);
    }

    @Override
    public ExtensionMenuProvider getInstance (JQuer
            IMenuManager menuContext) {
        return new RefactoringMenuProvider(view, me
    }

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    private void addTopLevelRefactorings (IMenuMana
        // can't do top queries if the tree isn't r
        if (view.isRootTree()) {
            IMenuManager topRefactorings = new Menu
            menu.appendToGroup(GROUP_REFACTORING, t

            try {
                JQueryRefactoringResults rs = JQuer

                for (Object result : rs) {
                    Map.Entry pair = (Map.Entry)res
                    Object[] parts = (Object[])pair
                    String[] path = (String[])parts
                    JQueryRefactoring refactoring =

                    Action action = new DoTopLevelR
                    createPath(topRefactorings, pat
                }
            } catch (JQueryException e) {
```

Move JQuery-based refactorings out into their own plugir

Prototype: two-column view



RefactoringMenuProvider.java

1.4 (awjb, 12-Aug-2008 5:11:59 PM)

```
public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTreeView.m

    private static final String REFACTORING_QUERY = "refactoring(!targets,?label,?
    private static final String[] REFACTORING_QUERY_CHOSENVARS = { "?label", "?que

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

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    @Override
    public void addAvailableMenu (IMenuManager menu,
            IStructuredSelection selection) {
        if ( ! (getElement(selection) instanceof QueryNode) ) {
            Object[] targets = findTargets(selection);

            if (targets.length > 0) {
                IMenuManager refactoringsMenu = new MenuManager("Refactor");
                nodeAddAvailableRefactorings(refactoringsMenu, targets);
                if (!refactoringsMenu.isEmpty()) {
```

* 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.
* "QueryRefactoring" TyRuBa type no longer used - its fields are merged into the single "refactoring" predicate we now use.

1.3 (awjb, 8-Aug-2008 12:19:13 AM)

```
public class RefactoringMenuProvider extends LayeredMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTreeView.m

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
        addTopLevelRefactorings(menu);
        addContextualRefactorings(menu, selection);
    }

    @Override
    public void addMenuGroup(IMenuManager menu) {
        menu.add(new Separator(GROUP_REFACTORING));
    }

    private void addTopLevelRefactorings (IMenuManager menu) {
        // can't do top queries if the tree isn't rooted at the factbase
        if (view.isRootTree()) {
            IMenuManager topRefactorings = new MenuManager("Available Top-Level Re
            menu.appendToGroup(GROUP_REFACTORING, topRefactorings);
```

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

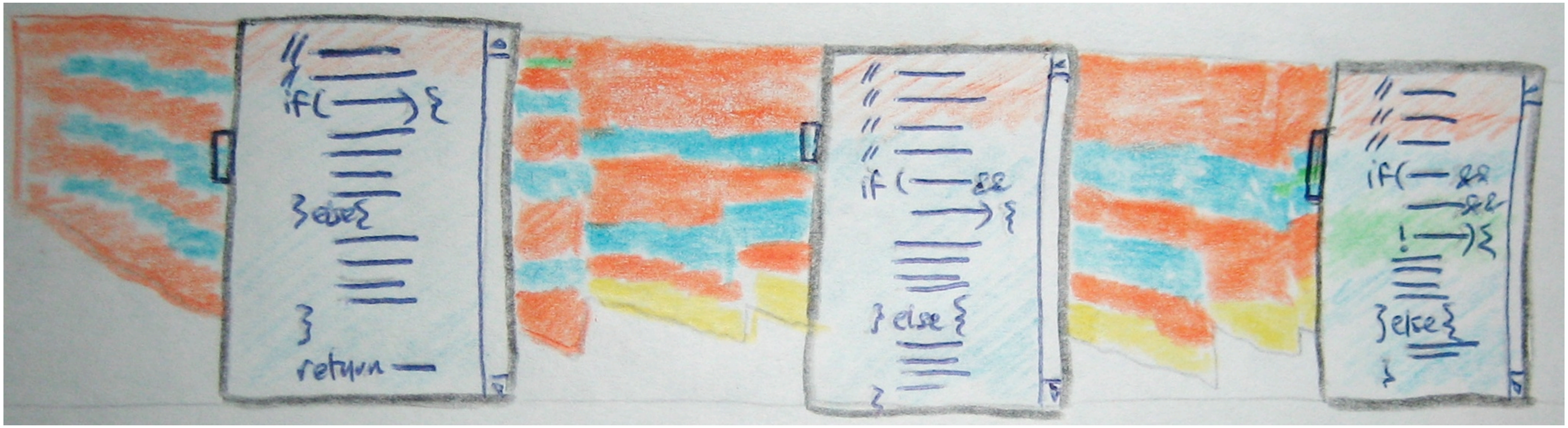
1.2 (awjb, 7-Aug-2008 10:58:25 PM)

```
public class RefactoringMenuProvider extends ExtensionMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTreeView.m

    public RefactoringMenuProvider () {
        super();
    }

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

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
```

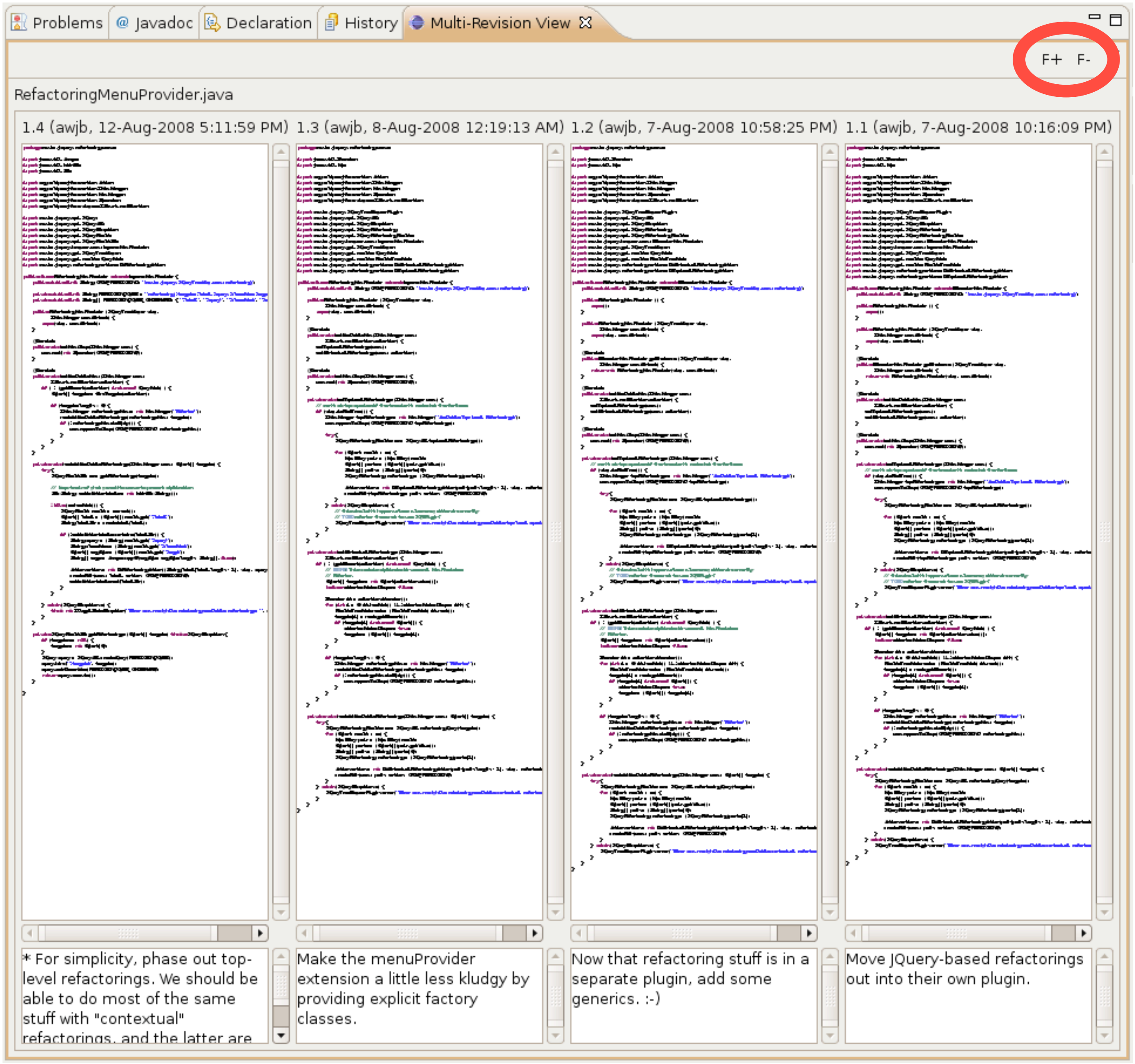
1.1 (awjb, 7-Aug-2008 10:16:09 PM)

```
public class RefactoringMenuProvider extends ExtensionMenuProvider {
    public static final String GROUP_REFACTORING = "ca.ubc.jquery.JQueryTreeView.m

    public RefactoringMenuProvider () {
        super();
    }

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

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

    @Override
    public void addAvailableMenu(IMenuManager menu,
            IStructuredSelection selection) {
```

Sketch: history flow



Prototype: “zoom” (change font size)



RefactoringMenuProvider.java

1.4 (awjb, 12-Aug-2008 5:11:59 PM) 1.3 (awjb, 8-Aug-2008 12:19:13 AM) 1.2 (awjb, 7-Aug-2008 10:58:25 PM) 1.1 (awjb, 7-Aug-2008 10:16:09 PM)

* 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

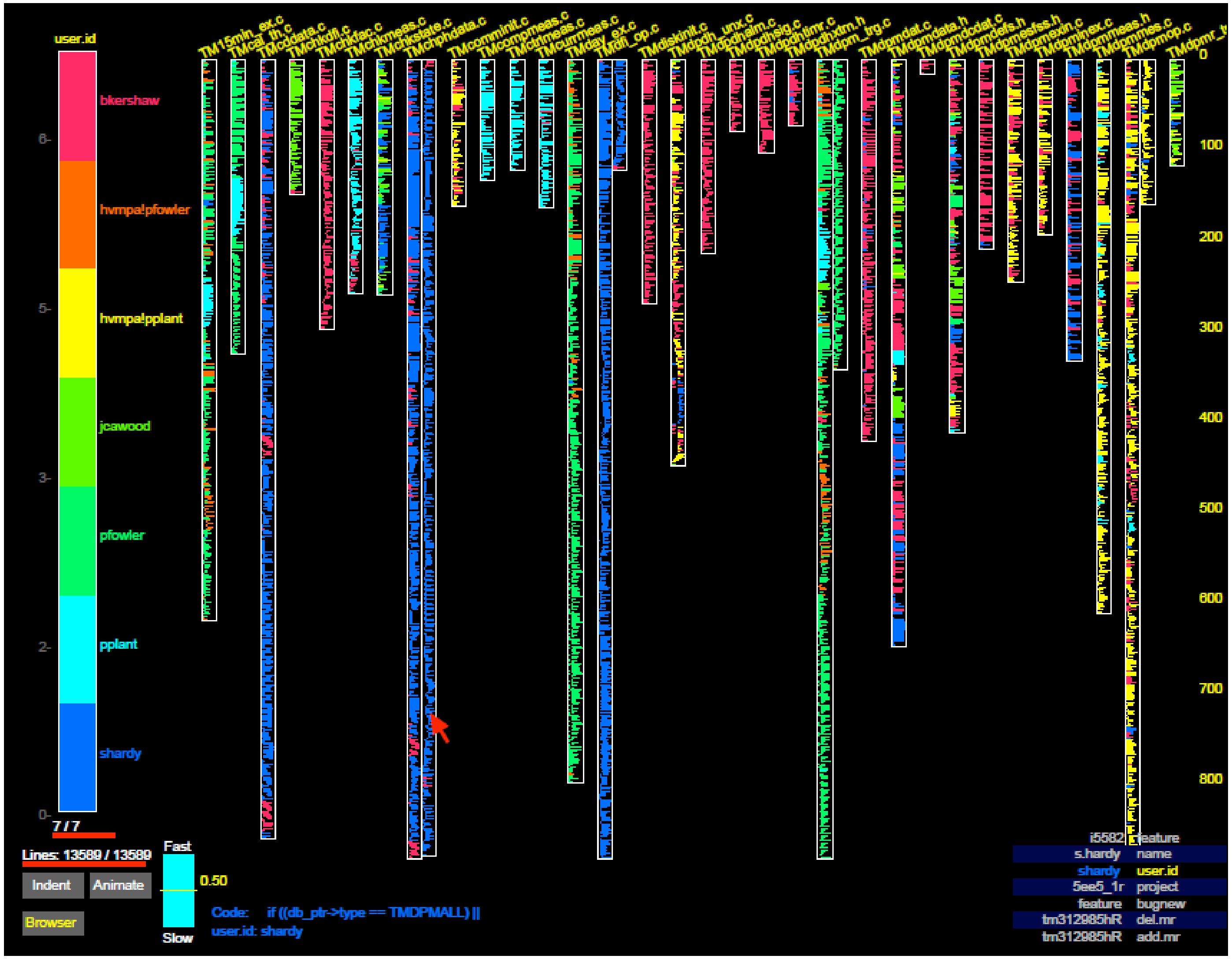
Make the menuProvider extension a little less kludgy by providing explicit factory classes.

Now that refactoring stuff is in a separate plugin, add some generics. :-)

Move JQuery-based refactorings out into their own plugin.

- At readable font sizes (~6-10), just helps to give more reading space
- When very small, gives coarse-grained visual overview
  - May provide intermediate point (in terms of “compression”) between full-size text and history flow?

Related Work: SeeSoft

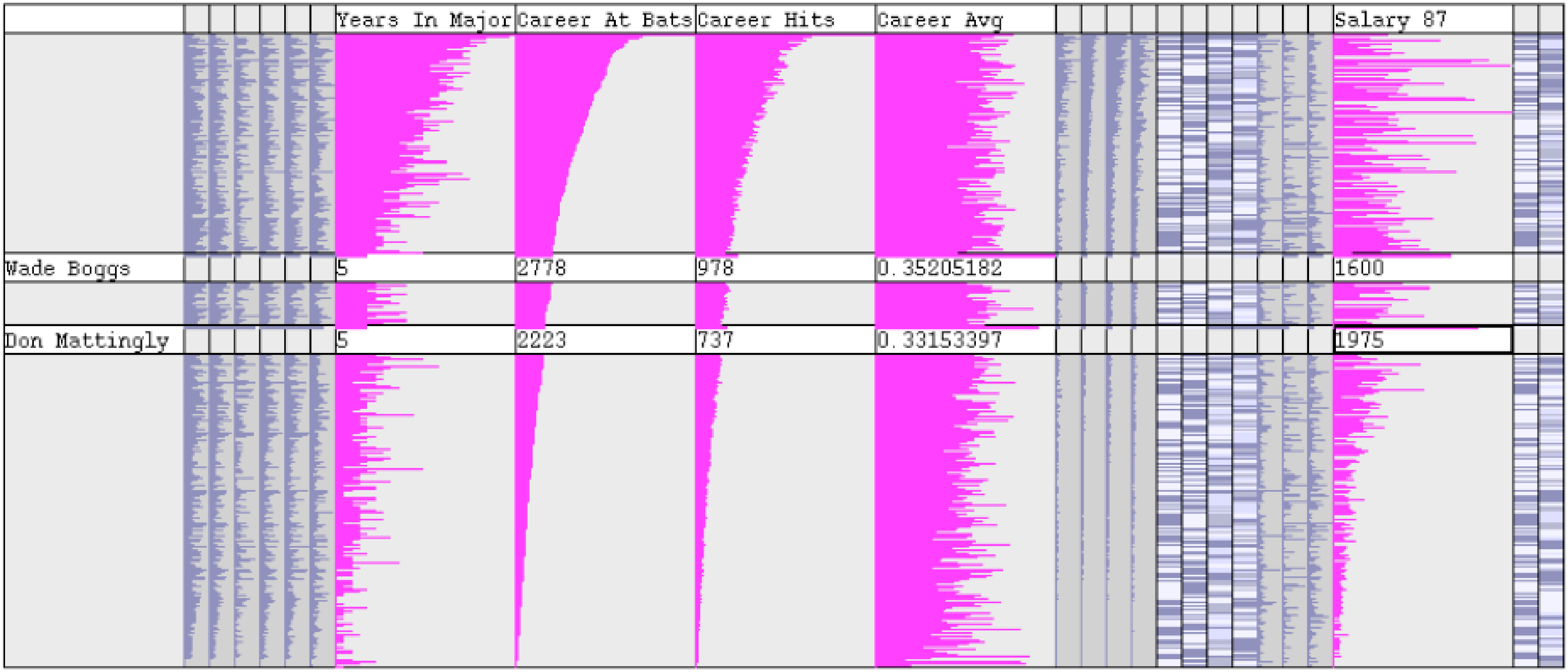


Code: if ((db_ptr->type == TMDPMALL) ||
user.id: shardy

i5582	feature
s.hardy	name
shardy	user.id
5ee5_1r	project
feature	bugnew
tm312985hR	del.mr
tm312985hR	add.mr

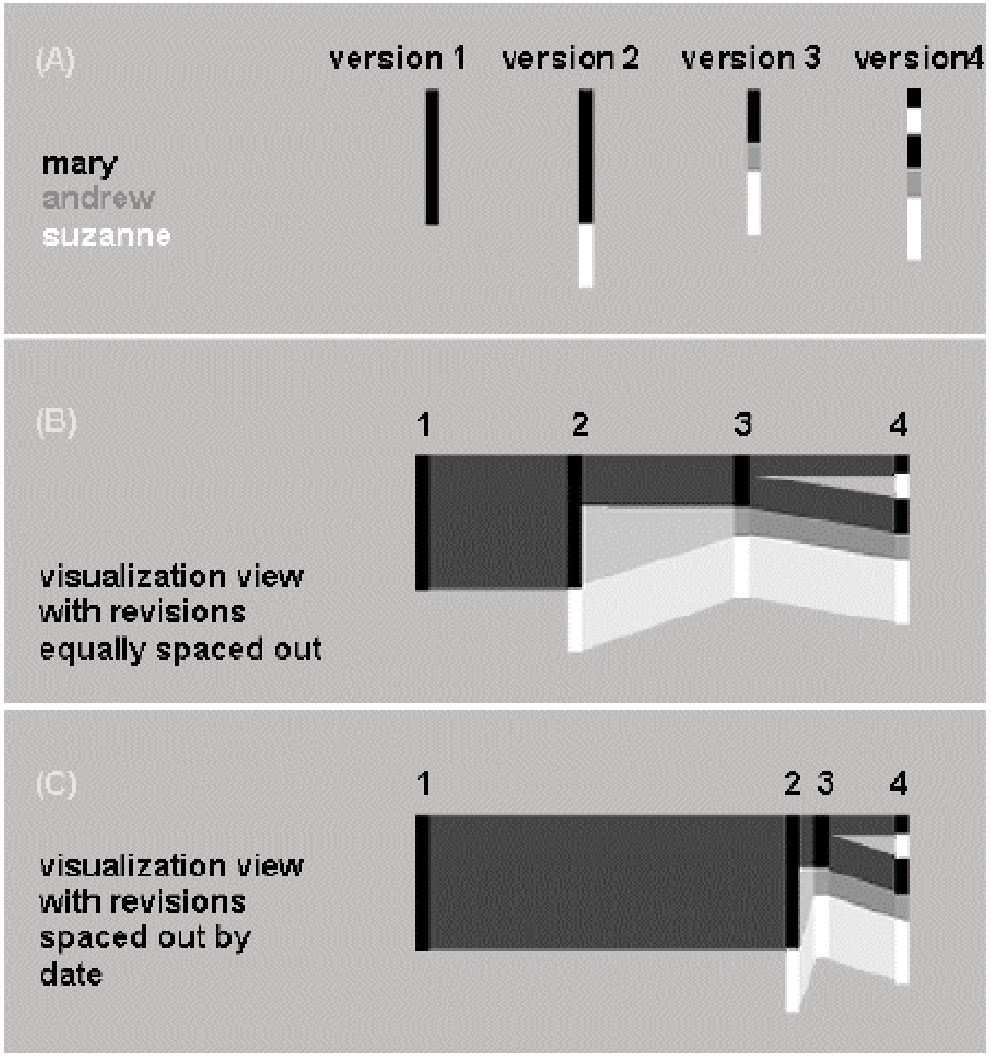
[Stephen G. Eick. Graphically Displaying Text. *Journal of Computational and Graphical Statistics*, Vol. 3, No. 2 (Jun., 1994), pp. 127-142. Figure 5. User IDs of Programmers Making Changes.]

Related Work: TableLens



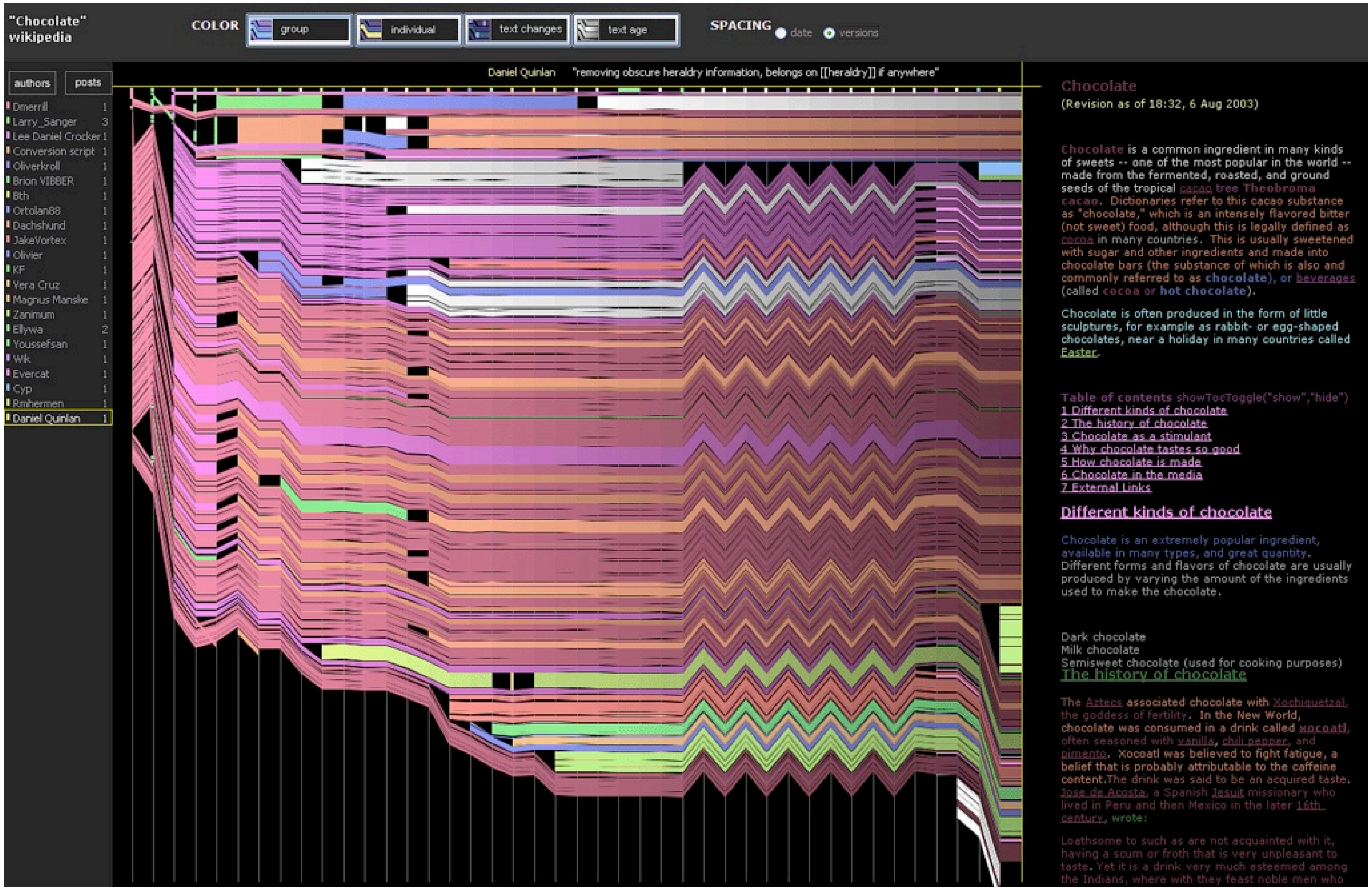
[Rao, R. and Card, S. K. 1994. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating interdependence* (Boston, Massachusetts, United States, April 24 - 28, 1994). B. Adelson, S. Dumais, and J. Olson, Eds. CHI '94.]

Related Work: History Flows


(A)
version 1
version 2
version 3
version4
mary
andrew
suzanne
(B)
1
2
3
4
visualization view with revisions equally spaced out
(C)
1
2
3
4
visualization view with revisions spaced out by date

[Viégas, F. B., Wattenberg, M., and Dave, K. 2004. Studying cooperation and conflict between authors with *history flow* visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04.]

Related Work: History Flows



authors	posts
Dmerrill	1
Larry_Sanger	3
Lee Daniel Crocker	1
Conversion script	1
Oliverkroll	1
Brion VIBBER	1
Bth	1
Ortolan88	1
Dachshund	1
JakeVortex	1
Olivier	1
KF	1
Vera Cruz	1
Magnus Manske	1
Zanimum	1
Ellywa	2
Youssefsan	1
Wik	1
Evercat	1
Cyp	1
Rmhermen	1
Daniel Quinlan	1

Daniel Quinlan "removing obscure heraldry information, belongs on [[heraldry]] if anywhere"

Chocolate

(Revision as of 18:32, 6 Aug 2003)

Chocolate is a common ingredient in many kinds of sweets -- one of the most popular in the world -- made from the fermented, roasted, and ground seeds of the tropical cacao tree Theobroma cacao. Dictionaries refer to this cacao substance as "chocolate," which is an intensely flavored bitter (not sweet) food, although this is legally defined as cocoa in many countries. This is usually sweetened with sugar and other ingredients and made into chocolate bars (the substance of which is also and commonly referred to as chocolate), or beverages (called cocoa or hot chocolate).

Chocolate is often produced in the form of little sculptures, for example as rabbit- or egg-shaped chocolates, near a holiday in many countries called Easter.

Table of contents showTocToggle("show","hide")
1 Different kinds of chocolate
2 The history of chocolate
3 Chocolate as a stimulant
4 Why chocolate tastes so good
5 How chocolate is made
6 Chocolate in the media
7 External Links

Different kinds of chocolate

Chocolate is an extremely popular ingredient, available in many types, and great quantity. Different forms and flavors of chocolate are usually produced by varying the amount of the ingredients used to make the chocolate.

Dark chocolate
Milk chocolate
Semisweet chocolate (used for cooking purposes)

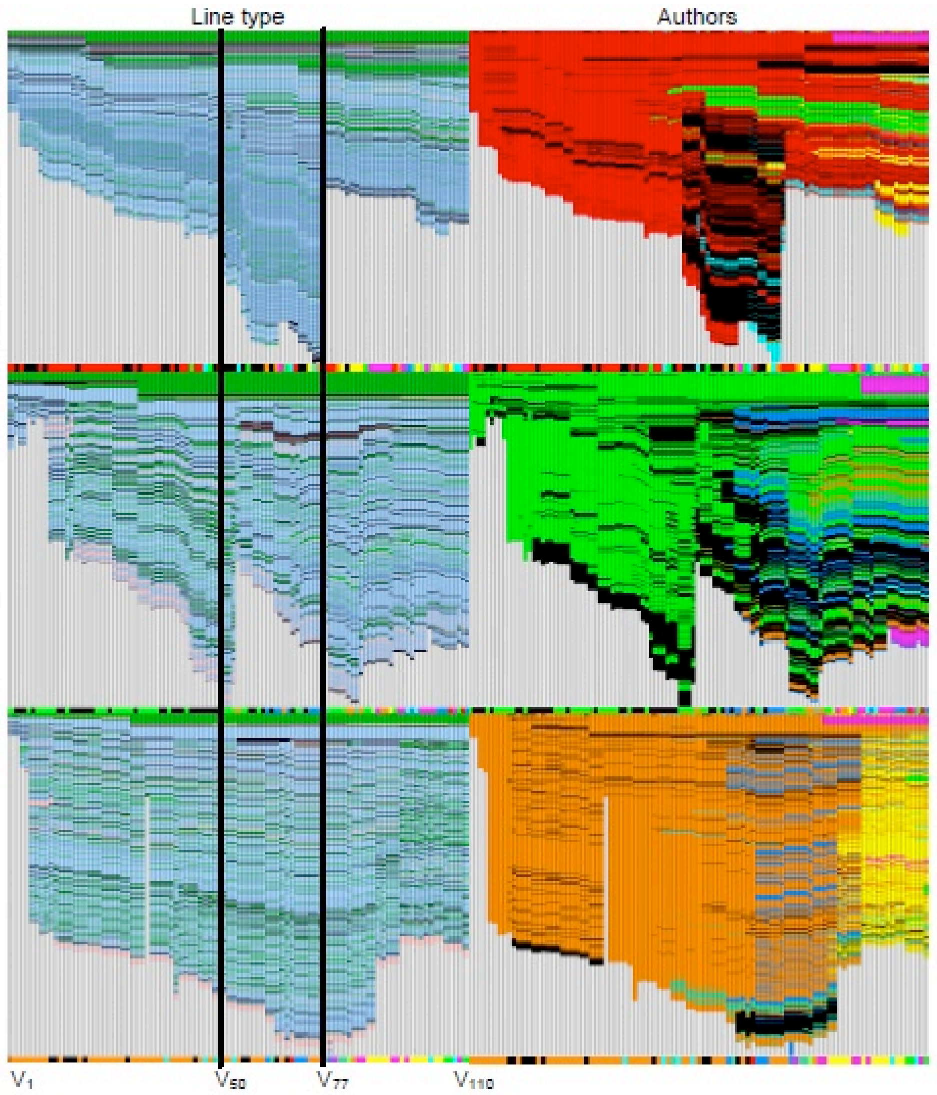
The history of chocolate

The Aztecs associated chocolate with Xochiquetzal, the goddess of fertility. In the New World, chocolate was consumed in a drink called xocoatl, often seasoned with vanilla, chili pepper, and pimento. Xocoatl was believed to fight fatigue, a belief that is probably attributable to the caffeine content.The drink was said to be an acquired taste. Jose de Acosta, a Spanish Jesuit missionary who lived in Peru and then Mexico in the later 16th century, wrote:

Loathsome to such as are not acquainted with it, having a scum or froth that is very unpleasant to taste. Yet it is a drink very much esteemed among the Indians, where with they feast noble men who

[Viégas, F. B., Wattenberg, M., and Dave, K. 2004. Studying cooperation and conflict between authors with *history flow* visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. Image source: Flickr user viegas]

Related Work: Visual Code Navigator



[Lommerse, G., Nossin, F., Voinea, L., and Telea, A. 2005. The Visual Code Navigator: An Interactive Toolset for Source Code Investigation. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (October 23 - 25, 2005). INFOVIS. Figure 8: Evolution view of three files, one on each row.]

Implementation Approach

- Initially considered standalone Java app
- Decided to implement as Eclipse plugin using Java SWT
  - Disadvantage: complexity of Eclipse plugin framework and SWT
  - Advantages:
    - Easy access to good Java CVS library (internal to Eclipse)
    - Easy access to other Eclipse features (e.g., syntax highlighting)
    - Integration with existing IDE makes it easy to apply to real code

Milestones

- **Friday, November 6:** Development environment set up; prototype capable of accessing CVS repository and downloading code. Small-multiples prototype started.
- **Monday, November 16:** Prototype of small-multiples views complete. Prototype of focus+context view started.
- **Wednesday, November 18:** Project update presentation.
- **Friday, November 27:** Prototypes of all views complete.
- **Thursday, December 10:** Final implementation complete. Final presentation and report drafted.
- **Monday, December 14:** Final project presentations.
- **Wednesday, December 16:** Final report submitted.

Milestones

- **Friday, November 6:** Development environment set up; prototype capable of accessing CVS repository and downloading code. Small-multiples prototype started. ✓
- **Monday, November 16:** Prototype of small-multiples views complete. ✓ Prototype of focus+context view started.
- **Wednesday, November 18:** Project update presentation. ✓
- **Friday, November 27:** Prototypes of all views complete.
- **Thursday, December 10:** Final implementation complete. Final presentation and report drafted.
- **Monday, December 14:** Final project presentations.
- **Wednesday, December 16:** Final report submitted.

Work Remaining

- Implement history flow view
  - See what types of colouring we can provide
  - Add corresponding colouring to small-multiples views
- Add graphical comparison-to-neighbours markings to current small-multiples views
- Other UI conveniences
  - Scrolling revision panes in tandem, resizing columns, suppressing revision notes, etc.
- Test with a realistic code task, find limitations of approach
  - Given limited screen space, how many revisions can we effectively compare at once?

Questions/suggestions?