Visualizing and Navigating Source Code History

Alex Bradley

CPSC 533C
University of British Columbia

December 14, 2009

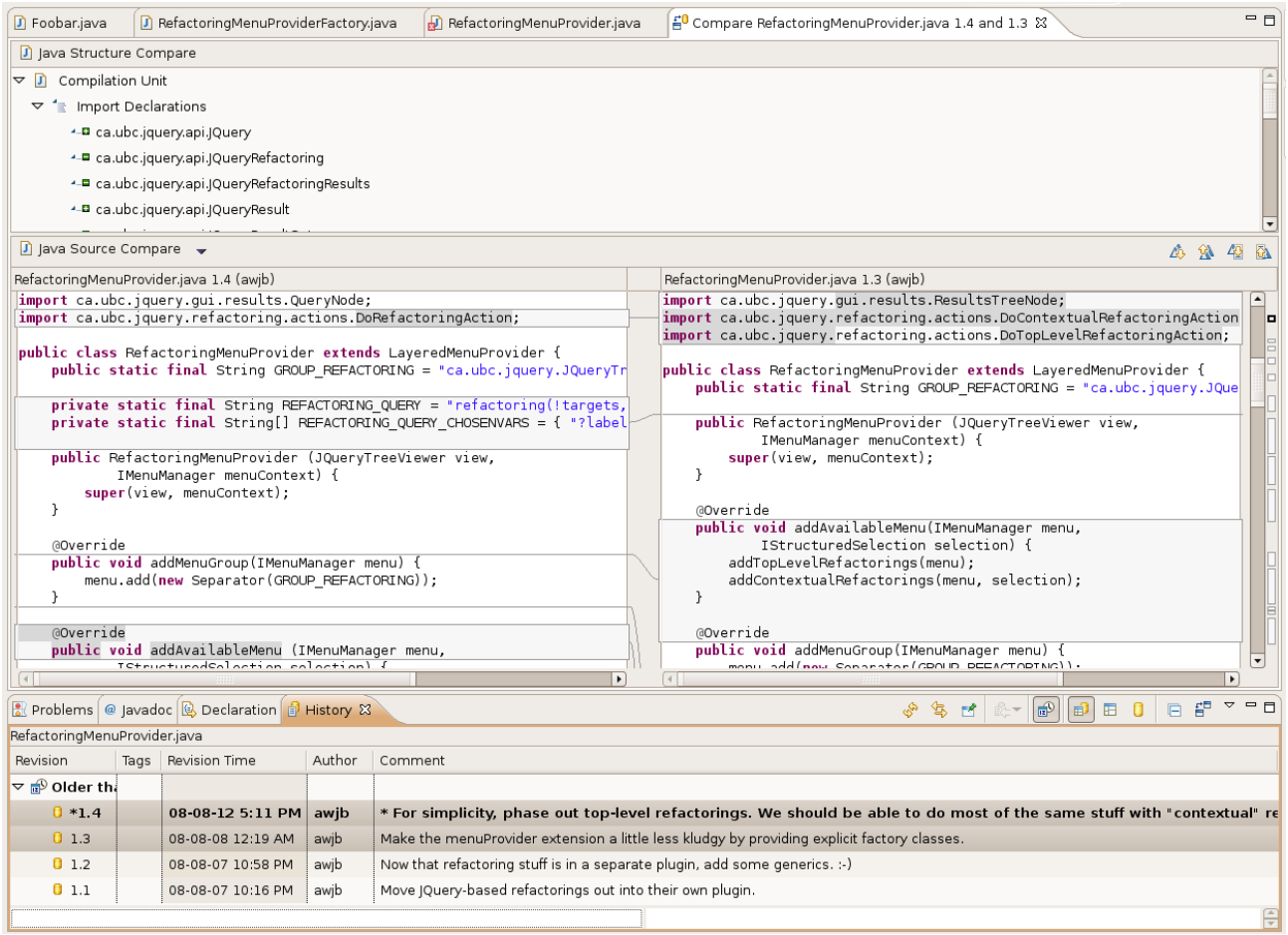
Outline

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

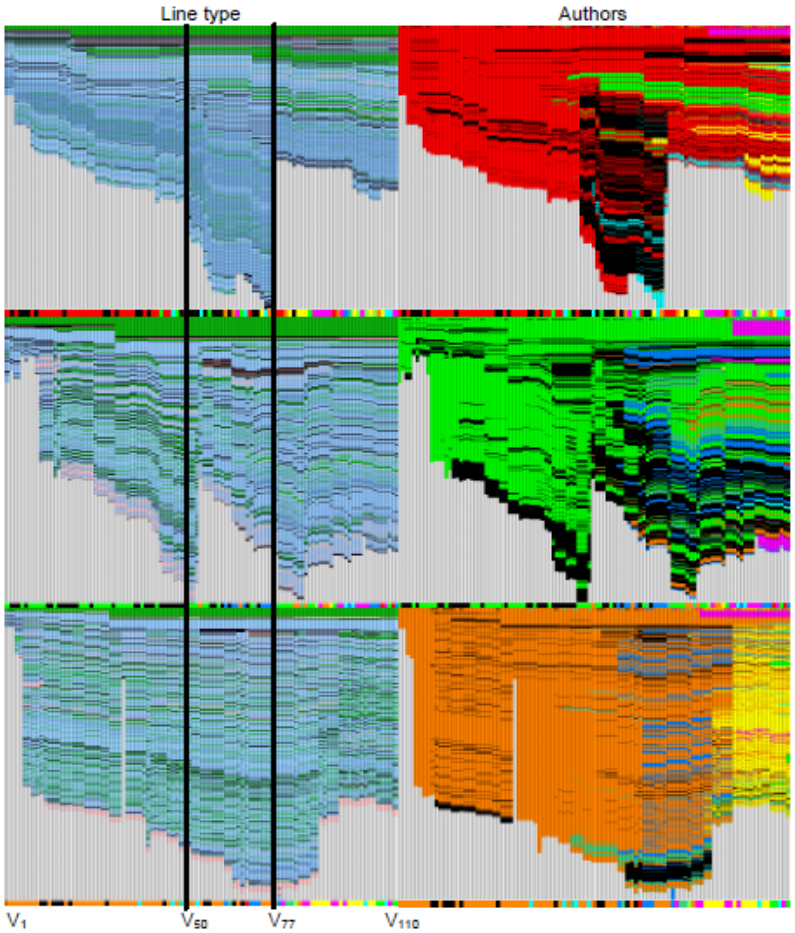
- Existing interfaces such as Eclipse revision control UI provide very good support for two-way revision comparison



Previous Work: Visual Code Navigator[1], CVSscan[2]

G. Lommerse[1], F. Nossin[1], L. Voinea[1,2], A. Telea[1,2], J. J. van Wijk[2]

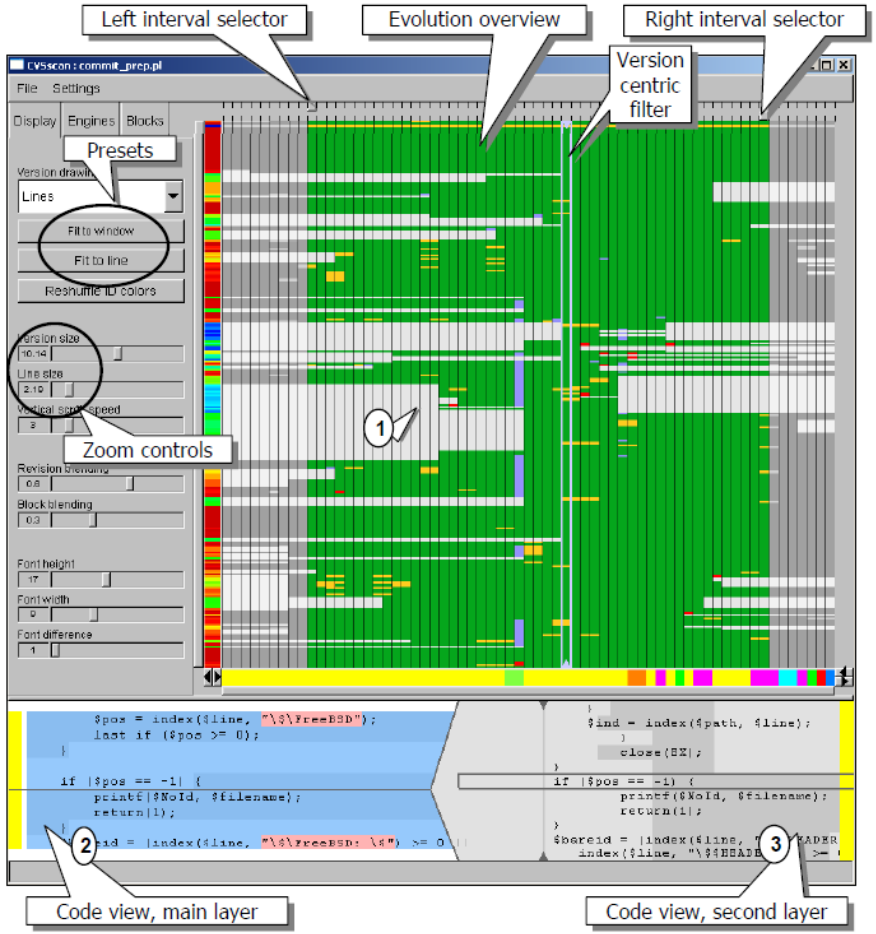
- Flow of vertical revision "stripes", with lines coloured according to some property of interest
  - Statement type
  - Author of line
  - Differences from other revisions
- Focused on "big picture" visualization; comparison of actual text of revisions less well supported



Previous Work: Visual Code Navigator[1], CVSscan[2]

G. Lommerse[1], F. Nossin[1], L. Voinea[1,2], A. Telea[1,2], J. J. van Wijk[2]

- 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

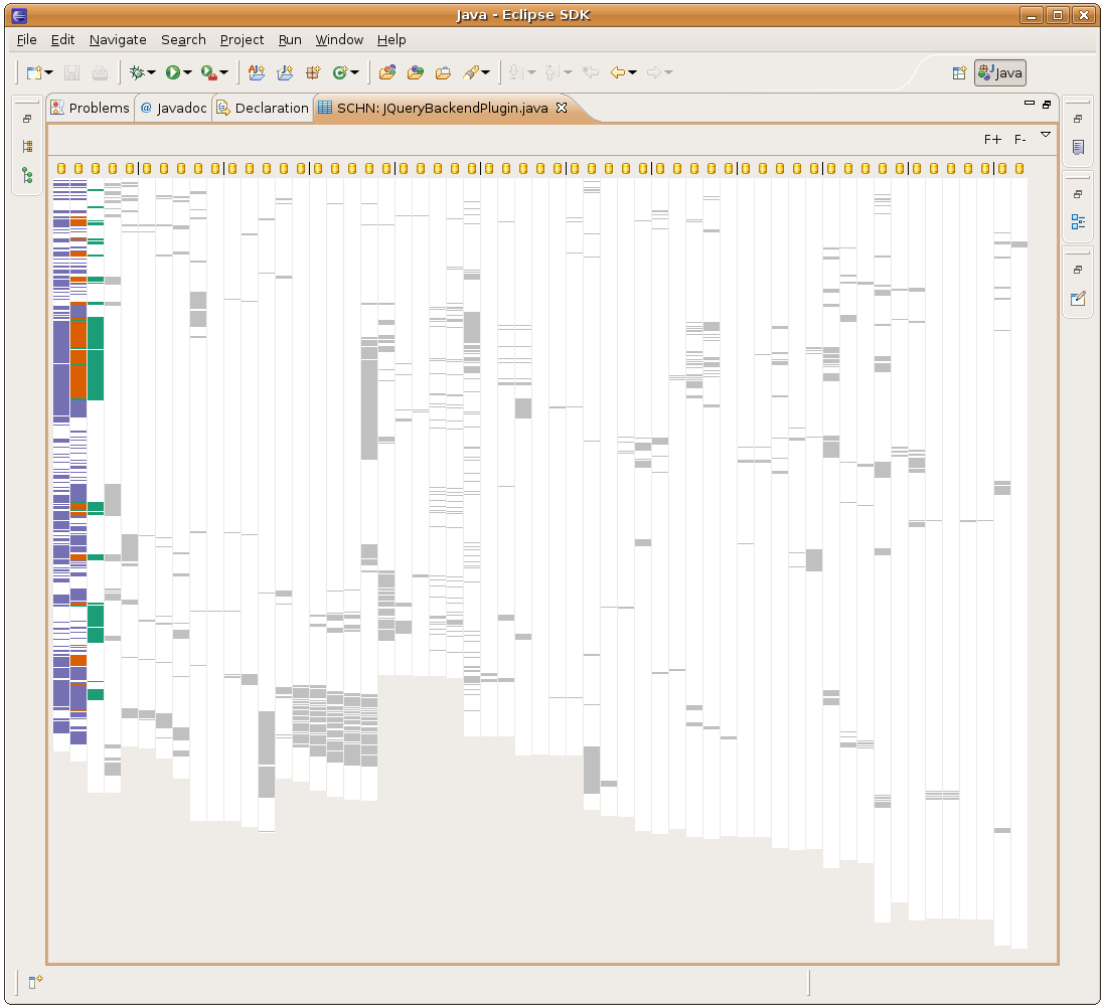


Project Approach

- 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

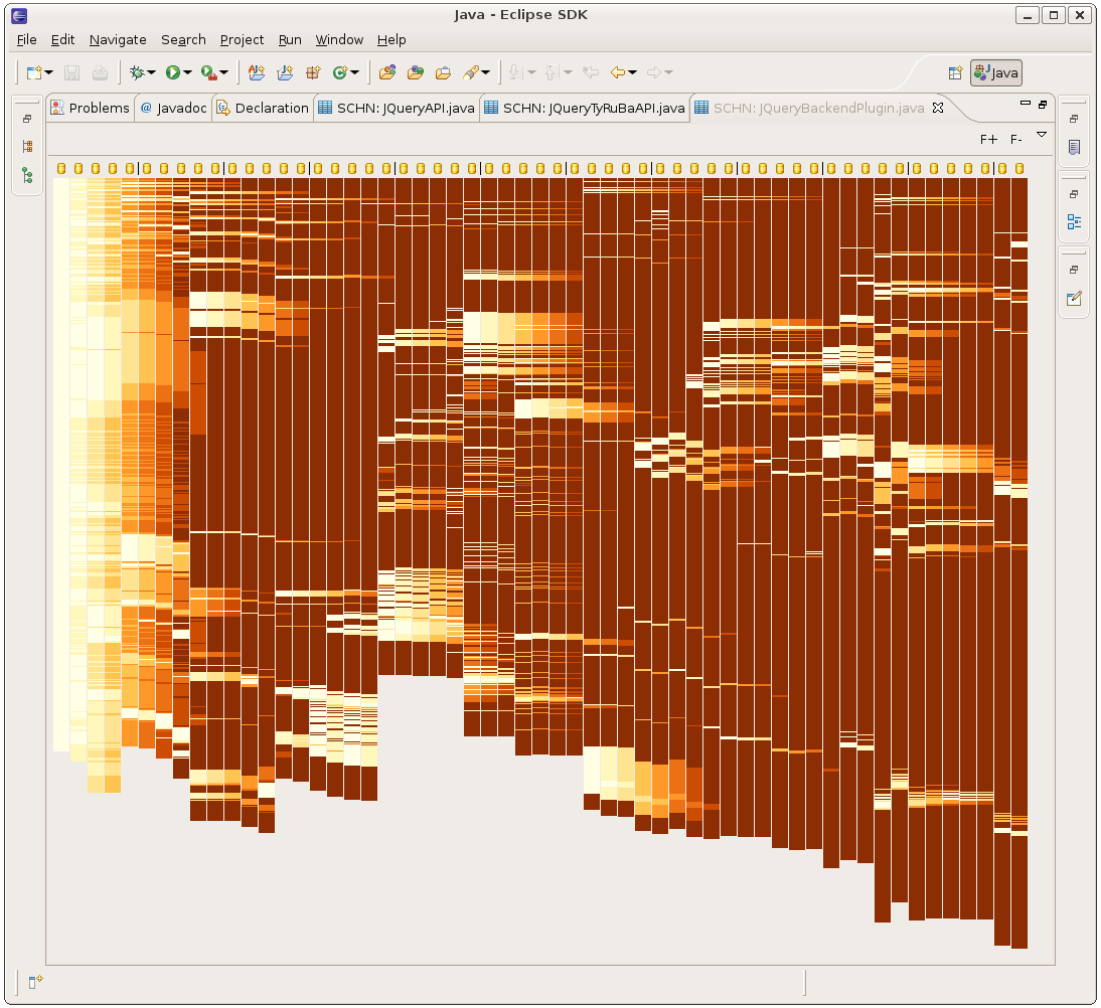
Overview: Revision Flow

57 revisions, lines coloured by differences from neighbouring stripes



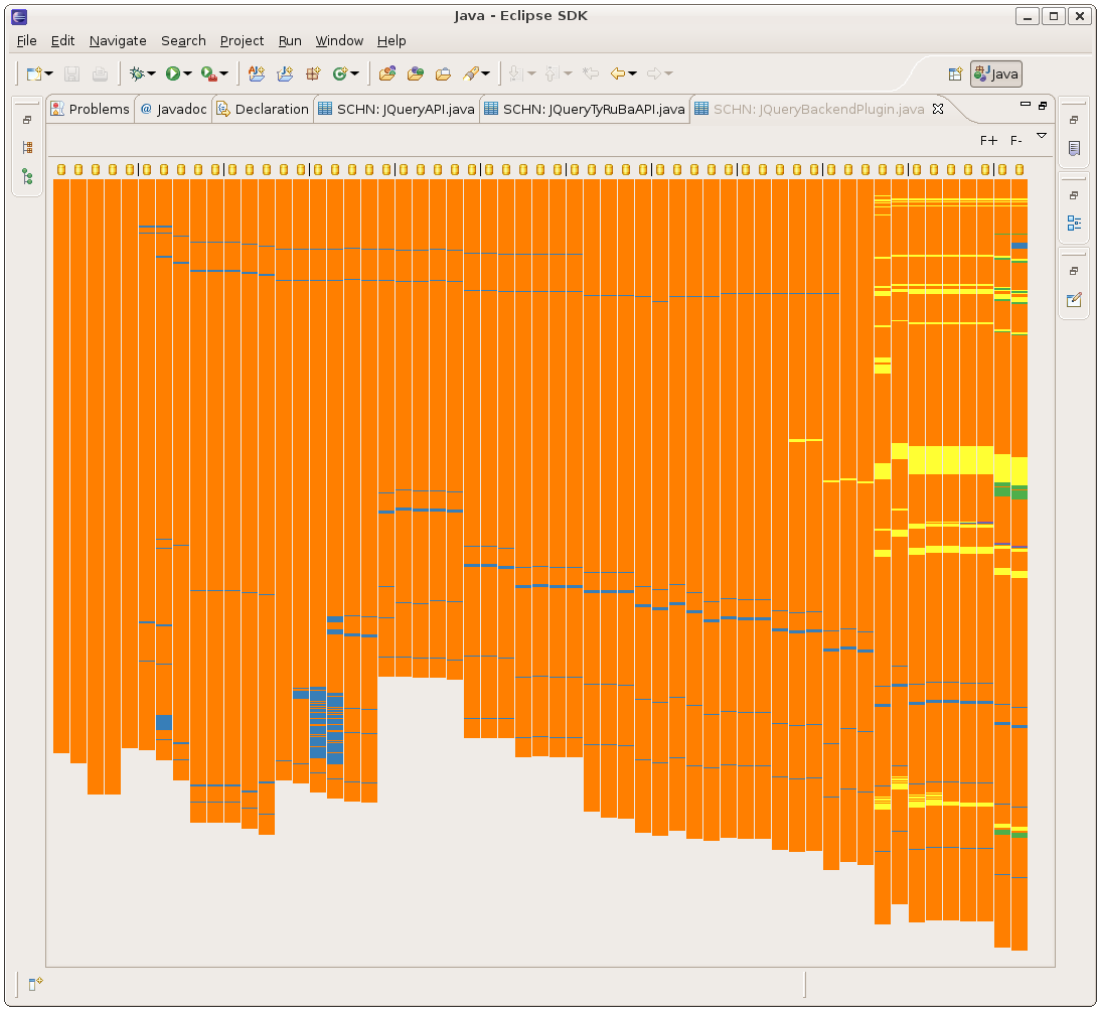
Overview: Revision Flow

57 revisions, lines coloured by code age



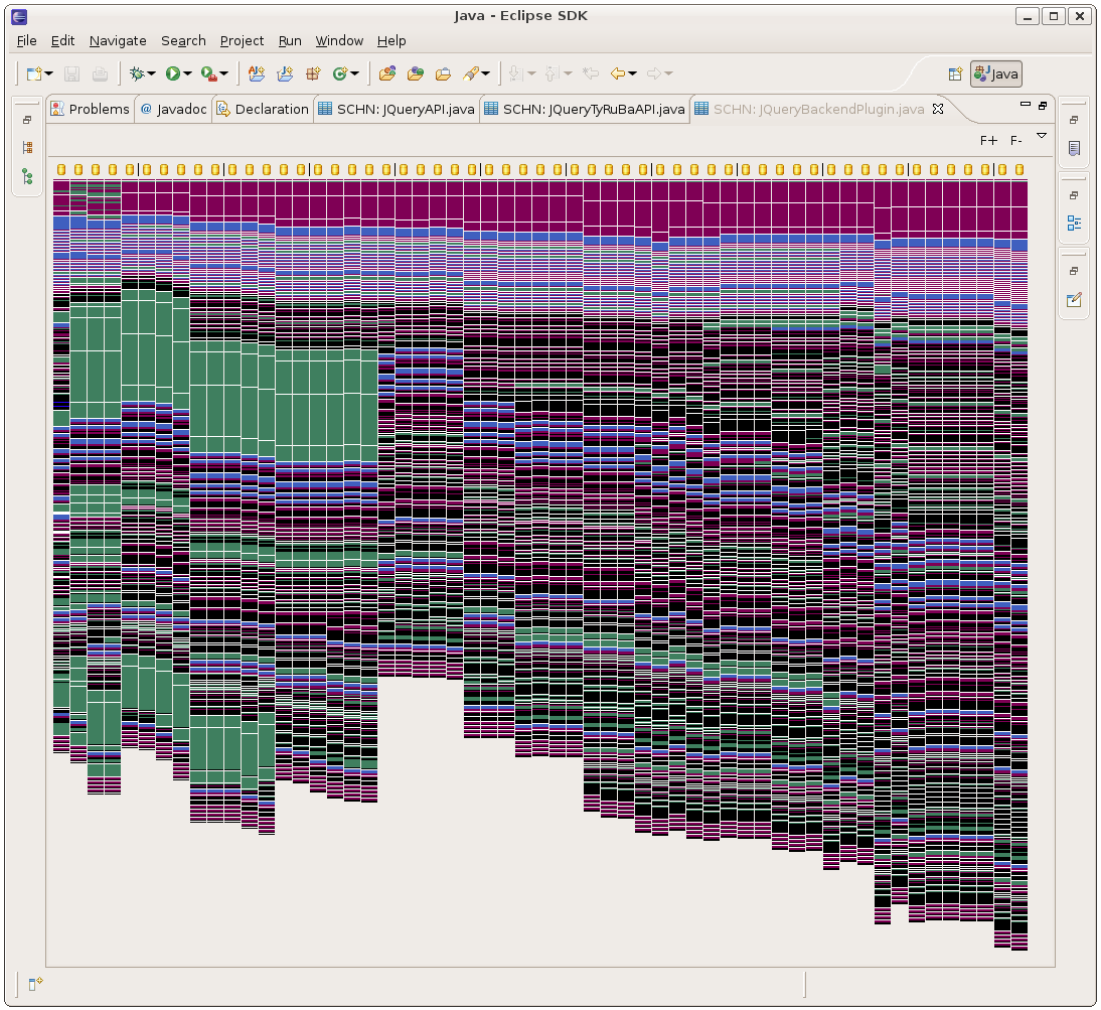
Overview: Revision Flow

57 revisions, lines coloured by authorship



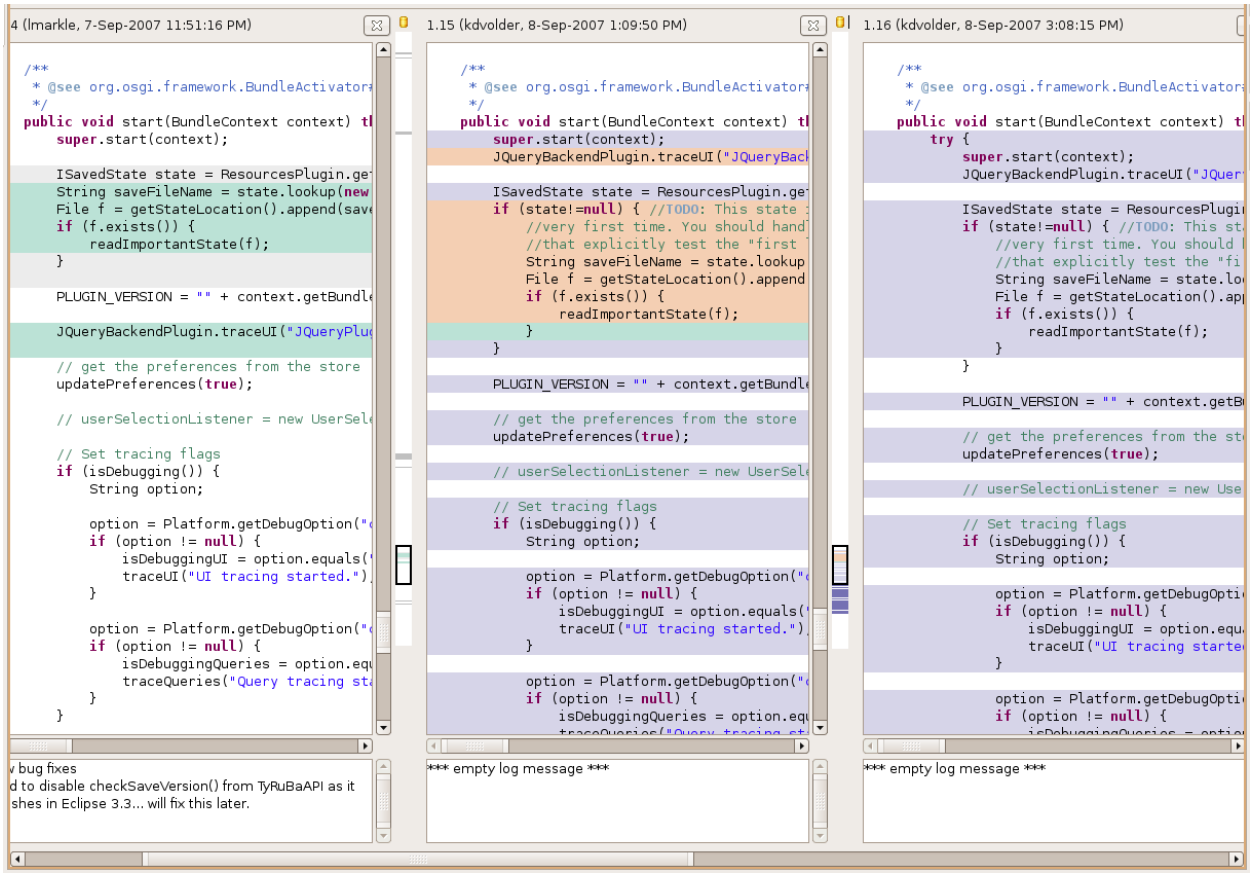
Overview: Revision Flow

57 revisions, lines coloured by statement type



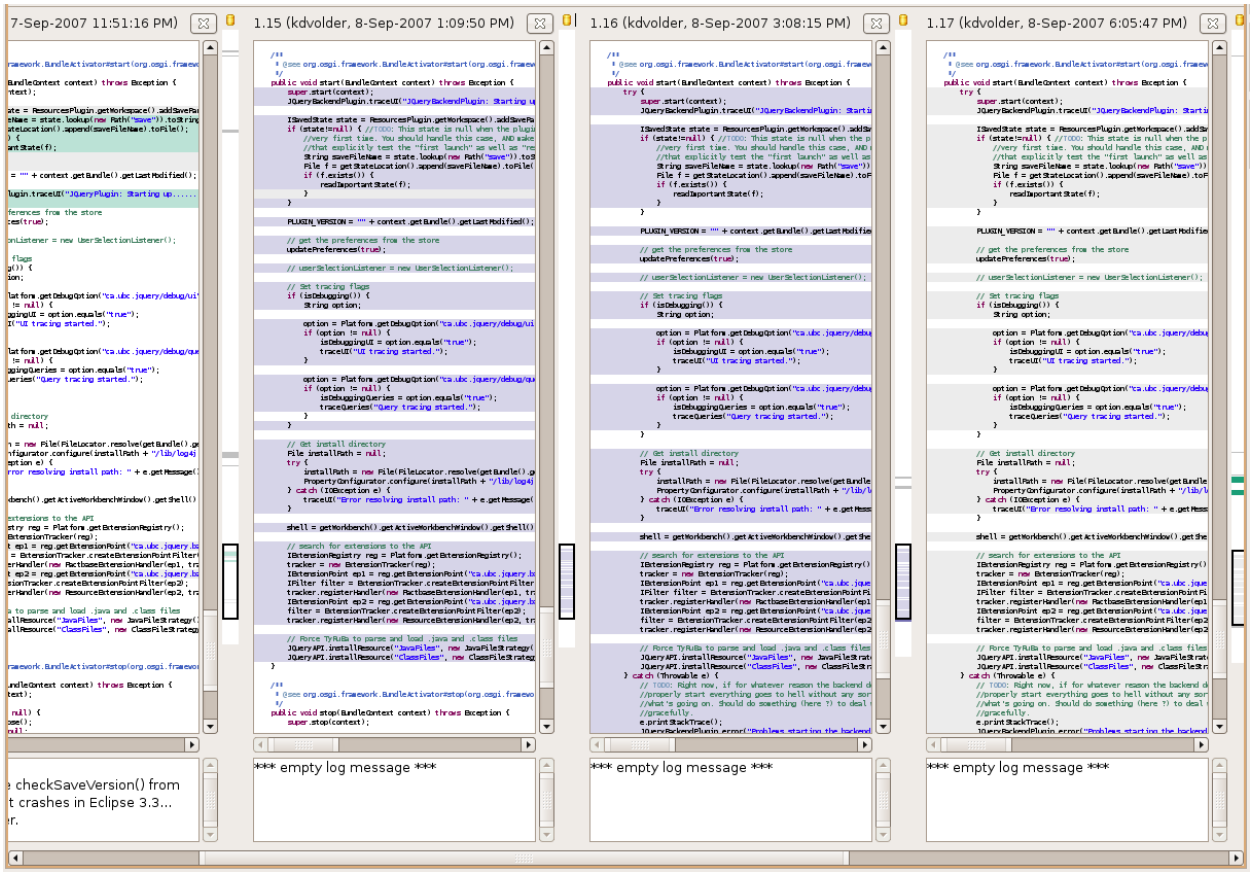
Detail: Many-revision comparison

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



Mid-level overview

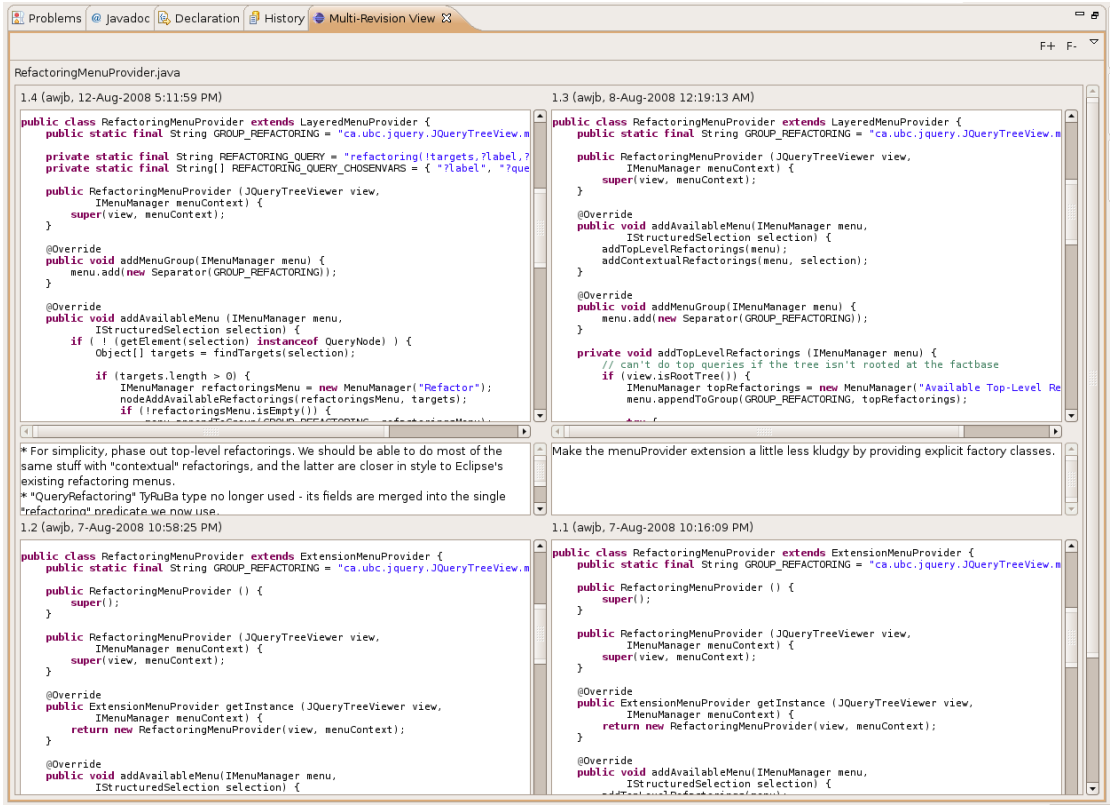
- Reducing font size in text viewers provide “middle ground” between stripe overview and full-size text detail



Demonstration

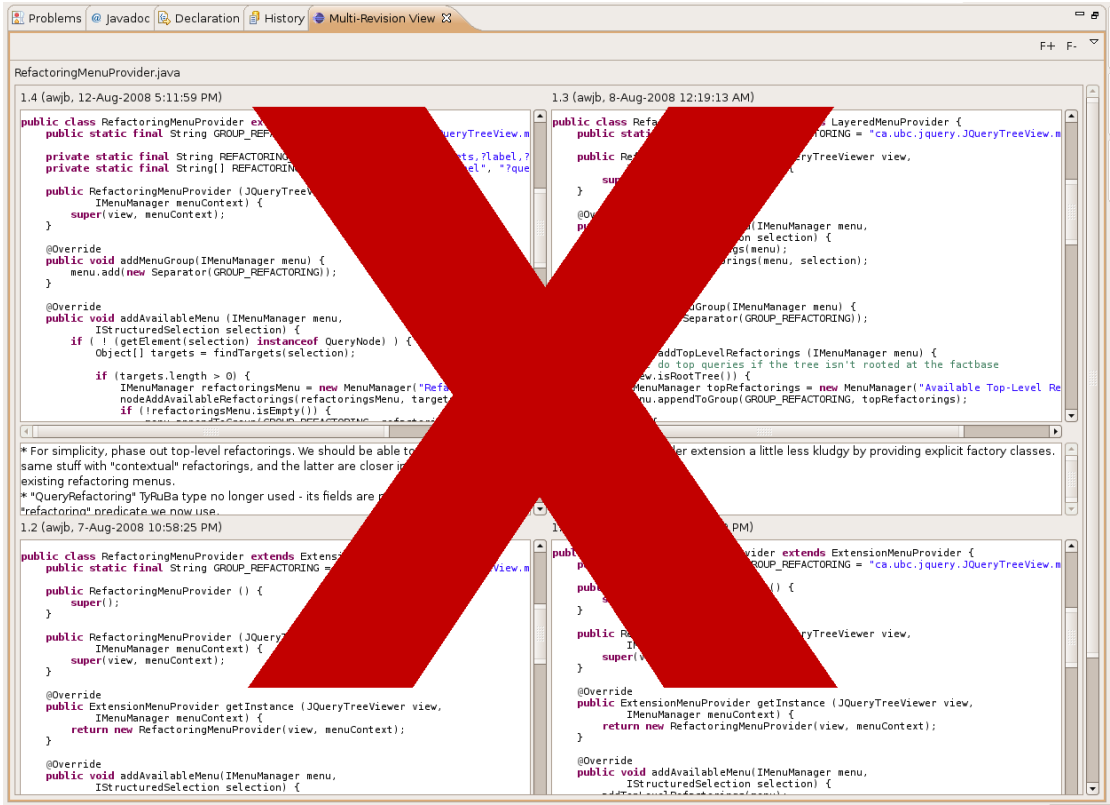
Lessons: Grid view not very useful

- 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



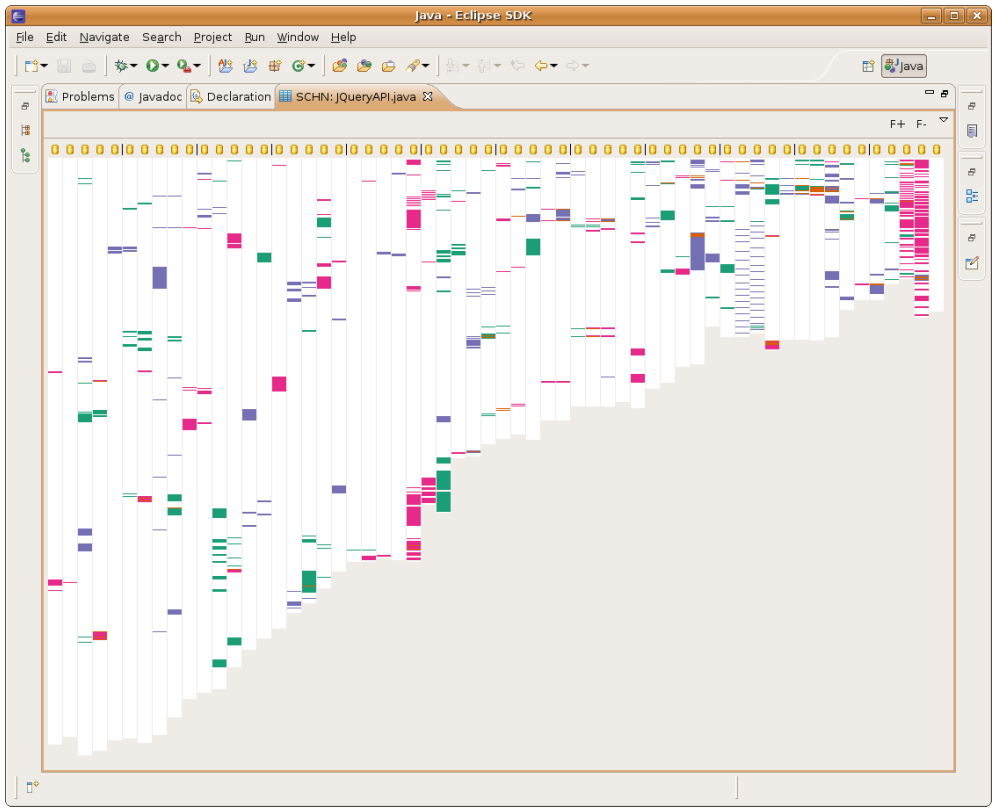
Lessons: Grid view not very useful

- 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



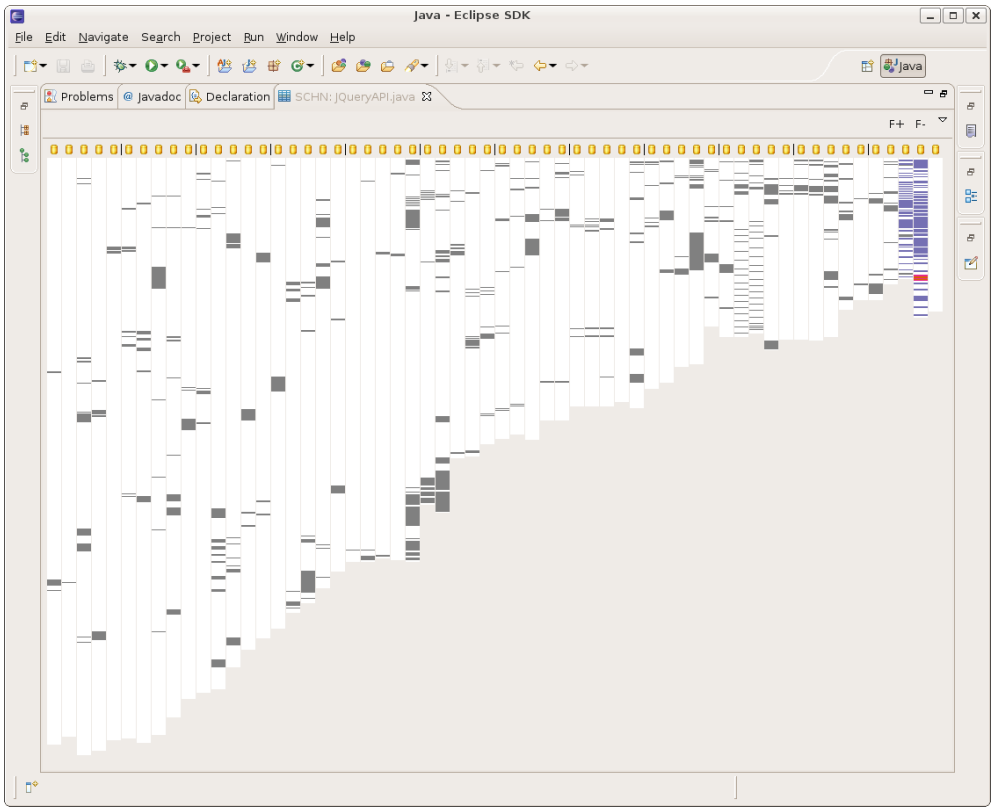
Lessons: Colours are tricky

- 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



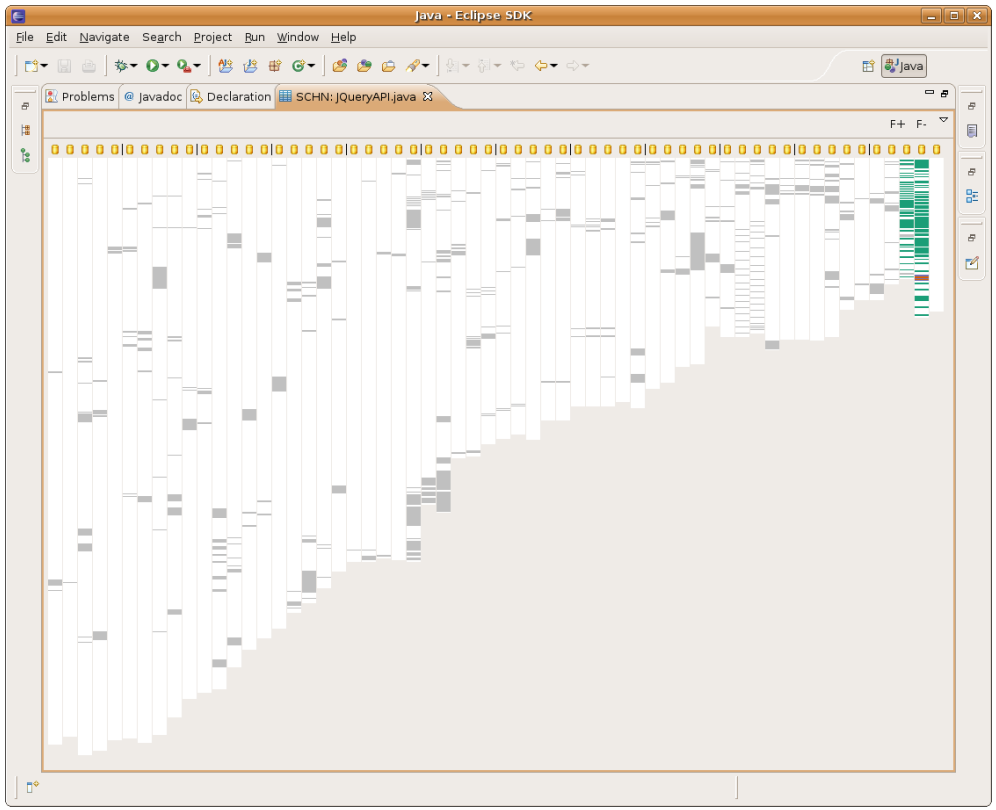
Lessons: Colours are tricky

- 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?