Mondrian: Code Review on the Web

Nov 30, 2006

Open Source Speakers Series

Guido van Rossum

guido@google.com

What is Code Review?

- When one developer writes code, another developer is asked to *review* that code
- A careful line-by-line critique
- Happens in a non-threatening context
- Goal is cooperation, not fault-finding
- Often an integral part of coding process
- *Involuntary* code review happens when debugging someone else's broken code
  - Not so good; emotions may flare

Benefits of Code Review

- Two pairs of eyes catch more bugs (duh)
  - Catch 'em early to save hours of debugging
- Enforce coding standards, style guides
  - Keep overall readability & code quality high
- Mentoring of new developers
  - Learn from mistakes without breaking stuff
- Establish trust relationships
  - Prepare for more delegation
- A good alternative to pair programming

Code Review in Open Source

- Author & reviewer on separate computers
- Author invokes "diff -u" to create patch file
- Author mails patch file to reviewer
  - or uploads to e.g. SourceForge patch manager
- Reviewer uses "patch" to recreate the files
- They email back-and-forth a few times
- Finally, *reviewer* submits into svn/cvs/etc
  - patch author often has no privileges (yet)
    - process helps "vetting" new developers

Google's Development Process

- (In theory :-)
- Single company-wide Perforce (p4) depot
  - No developer branches
- Company-wide NFS
  - Developers workspaces accessibly by others
- All code reviewed **before** submission
  - All code-review email logged for auditors
- Wrapper "g4" implements superset of "p4"
  - Originally a shell script; now a Python program

Google's Code Review Process

- All command-line and email based:

  1. Author edits changes in workspace, tests etc.
  2. Author send email to reviewer (a tool helps)
  3. Reviewer views the diff (another tool helps)
  4. Reviewer sends mail back (regular mail reply)
  5. Rinse and repeat (using regular mail replies)
  6. When reviewer replies "lgtm", author submits
     - lgtm = looks good to me

Original, Minimal Tools

- "g4 change" invoked by author defines set of files involved, adding comments
- "g4 mail" sends form email to reviewer
  - (also integrated with g4 change call)
- "g4 diff" invoked by reviewer diffs files straight out of author's workspace
  - "tkdiff" X11-based side-by-side differ
- "g4 reply" synthesizes reply mail
  - optional; not used very much

Problems With Old Process

- VPN usage (reviewers working from home)
  - can’t use tkdiff, or it is too slow
- Line numbers change as code evolves
  - line-oriented comments get out of sync
- Can't diff between pre-submit revisions
  - reviewer must keep manual track of evolution
- Email can get lost in inbox; no work flow
  - reviewer: what do I still need to review?
  - author: is my code reviewed?

Enter Mondrian

- Web-server based
  - Addresses VPN issues
  - Snapshots all previously reviewed versions
- Shows side-by-side diff (like tkdiff)
- Lets you add *in-line* comments
  - Collect comments into a single message
- Also receives and organizes email
  - Not all parties need to use Mondrian

Live Demo

- Show dashboard
  - explain categories; hover over CL, flags, users, description
- Show someone else’s dashboard
- Show a group’s TO DO list (build-grouplet)
- Show a CL view
  - browse some comments (expand/collapse)
  - follow links to Sourcerer, p4web
  - show Request code review form; Upload snapshot form; Approve form
- Show a file view
  - use n/p to navigate diff chunks
  - navigate between files
  - show intra-region diffs
  - add, edit, discard, reply to in-line comments
- Show Review and publish form

How It Started

- As a Noogler, I needed a starter project
- Someone proposed a code review web app

- Decided to freshen up my web knowledge
- Learned some Django on the way
- Adopted WSGI (PEP 333) while I was at it
- Learned about Google's Bigtable and p4lib
- Early prototype was immediately hot!

How It Works (Overview)

- Bigtable (a Googly db) used to store:
  - Change metadata (description, list of files, …)
  - Comments (entered on web or received email)
  - File snapshots taken from user workspaces
  - Per-user data (active changes, last view dates)
- Web server talks to user, Bigtable, p4
  - Also to NFS, SSH when fetching snapshots
- Mail server processes incoming email
  - Storing them as comments

Snapshots

- Copies of files taken from user's directory
  - Updated whenever change view visited
  - Solve two problems with the old process/tools
- What if the files aren't on NFS?
  - Linux: use SSH/SCP to user's workstation
  - Others: author uploads plain old patch
- Snapshots and comments are kept forever
  - To satisfy auditors
  - To track extended history of changes

Web User Interface

- Mostly pretty traditional HTML with CSS
  - Django templates, wsgiref server (all Python)
- Some JavaScript for UI niceties
  - Expand, collapse comments
  - Show unified diffs in-line in CL view (AJAX)
- Full file view uses more AJAX:
  - Keyboard navigation
  - Inline draft comment editing
    - Avoids full-page refresh

Editing Inline Draft Comments

- Each line in the diff is two table rows:
  - <tr><td> 123 textA</td><td> 127 textB</td></tr>
  - <tr><td>…</td>   <td>…</td>   </tr>
  - First row displays left & right diff text
  - Next row (hidden) used for inline comments
  - Each <td> has a unique id {old,new}+lineno
- Double click on first row calls JavaScript
  - Inserts form into next row
  - “Submit” sends form data to server
  - Server sends back HTML to replace form

Mail Server

- Receives every email sent to review logs
- Ignores messages sent by Mondrian itself
- Parses subject looking for revision #
- Inserts message as comment into Bigtable
- Also updates reviewers’ TO DO queues

- Implemented using standard smtpd.py

Performance

- Mostly fast enough, on just one server
- Most of the work is done elsewhere:
  - Bigtable+GFS server infrastructure
  - Perforce server
  - In the browser (rendering reams of HTML)
- What's slow:
  - Contention for Perforce
  - Generating HTML for loaded dashboard
  - Diffing huge files (uses Python's difflib)