Engineering Design w/Embedded Systems

Lecture 30—Reviews (including Code Reviews)

Patrick Lam
University of Waterloo

March 25, 2013

Part I

Reviews

On Reviews

Review:
activity where reviewers examine a work product to provide feedback.

Advantage:
reveal defects early—defects less costly to fix.

What to review:
requirements specifications; schedules; bug reports; design documents; **code**; test plans; test cases.

Informal vs Formal

- *informal review*: written or verbal review requested by a developer of a work product.
- *formal review*: written review conducted by a team leader or a moderator to identify, document, and fix defects in a work product.

Types of Reviews

- **desk check**: informal review;
  author distributes work to peers for reviews and comments.
- **walkthrough**: informal review meeting;
  moderated by the author.
- **inspection**: formal review meeting;
  guided by a moderator.
  Produces a log of identified defects in a work product.
- **code review**: software inspection
  identifying, logging, and perhaps correcting bugs.

Desk Checks

Desk check: first line of defence against defects.

- can speed up formal inspections
  by taking care of simple defects in desk checks first.
- for many work products, desk checks suffice;
  often don't need a formal inspection.

But:

- only effective if taken seriously.
- easy to just say "LGTM" (Looks Good To Me)
  without actually checking the product.

  It's important to spend enough time on desk checks, and managers must allocate time for them.

Walkthroughs

Walkthrough: guided review of a work product.

- allow people with less expertise to review a work product;
- users of the work product often invited to walkthroughs.
- New points-of-view often help identify defects.

Author of the work product presents the design and ensures that the attendees understand its design.

How a walkthrough works:

- Before: distribute presentation materials.
- During: solicit feedback from the audience.
- After: follow up with attendees who have helped out by giving comments.

Inspection

Inspection: formal review meeting where participants identify and document defects or possible improvements.

Participants identify, and propose solutions to, defects.

A good mix of participants (but not too many!) helps find previously-overlooked defects (subtle, complex bugs).

Steps in a formal inspection

- **Preparation**: Before the meeting, distribute the work project to each member of the inspection team, plus a checklist indicating what to review.
- **Overview**: Moderator provides an overview of the item.
- **Page-by-page Review**: Moderator walks the inspection team through the work product and logs defects.
- **Rework**: Afterwards, the author goes through the list of defects and fixes them.
- **Follow-up**: Inspection team members verify that the author has fixed the defects.
- **Approval**: Inspection team approves the work.

We do something similar for master's and PhD theses.

Code Review

Code review: examines source code (usually a patch) to identify defects or possible improvements.

Coverage options:

- review everything (Mozilla); or,
- review a representative sample.

Representative sample:

developers tend to repeat the same mistakes.

If you find one bug, look for similar ones nearby.

Code Review (sampling)

When sampling, here are some places to look:

- source code that only one person has the expertise to maintain;
- tricky algorithms that are susceptible to defects;
- source code that calls difficult-to-use libraries;
- code written by inexperienced developers; and
- functions that could fail catastrophically if a defect is present.

Code Review (who? what?)

In industry: by team?

Open-source world: 1 or 2 experienced developers, independently.

What to look for:

- clarity, maintainability, accuracy, reliability,
- robustness, security, scalability, reusability, efficiency.

Pair Programming

Part of Extreme Programming.

Can also serve as instant code review.

Part II

Reviews for Open-Source Projects

How Open-Source Projects Work

Typically:

- an official repository. (SVN, Git, Mercurial)
- a set of committers, who may commit changes.

Outside contributors: may send patches
(bug fixes, new features).

- a committer reviews the patch before committing it.

Committers may/must also seek review for their patches.

Case Study: Reviews at Mozilla

Mozilla Foundation: develops the Firefox web browser (and other projects).

Case Study: Reviews at Mozilla

Huge codebase $\Rightarrow$ elaborate reviewing policy[1].

- require at least one review ("owner/peer review") for all patches, plus
- second review ("super-review") for many patches.

[1] http://www.mozilla.org/hacking/reviewers.html, https://developer.mozilla.org/en/Code_Review_FAQ

Case Study: Peer review at Mozilla

Owner/peer review: by a domain expert who understands the code being modified and the implications of the change.

> *A review is focused on a patch's design, implementation, usefulness in fixing a stated problem, and fit within its module.*

Reviews check for:

- whether the patch fixes a problem;
- API/design;
- maintainability;
- security;
- integration;
- testing; and
- license compliance.

Case Study: Super-reviews at Mozilla

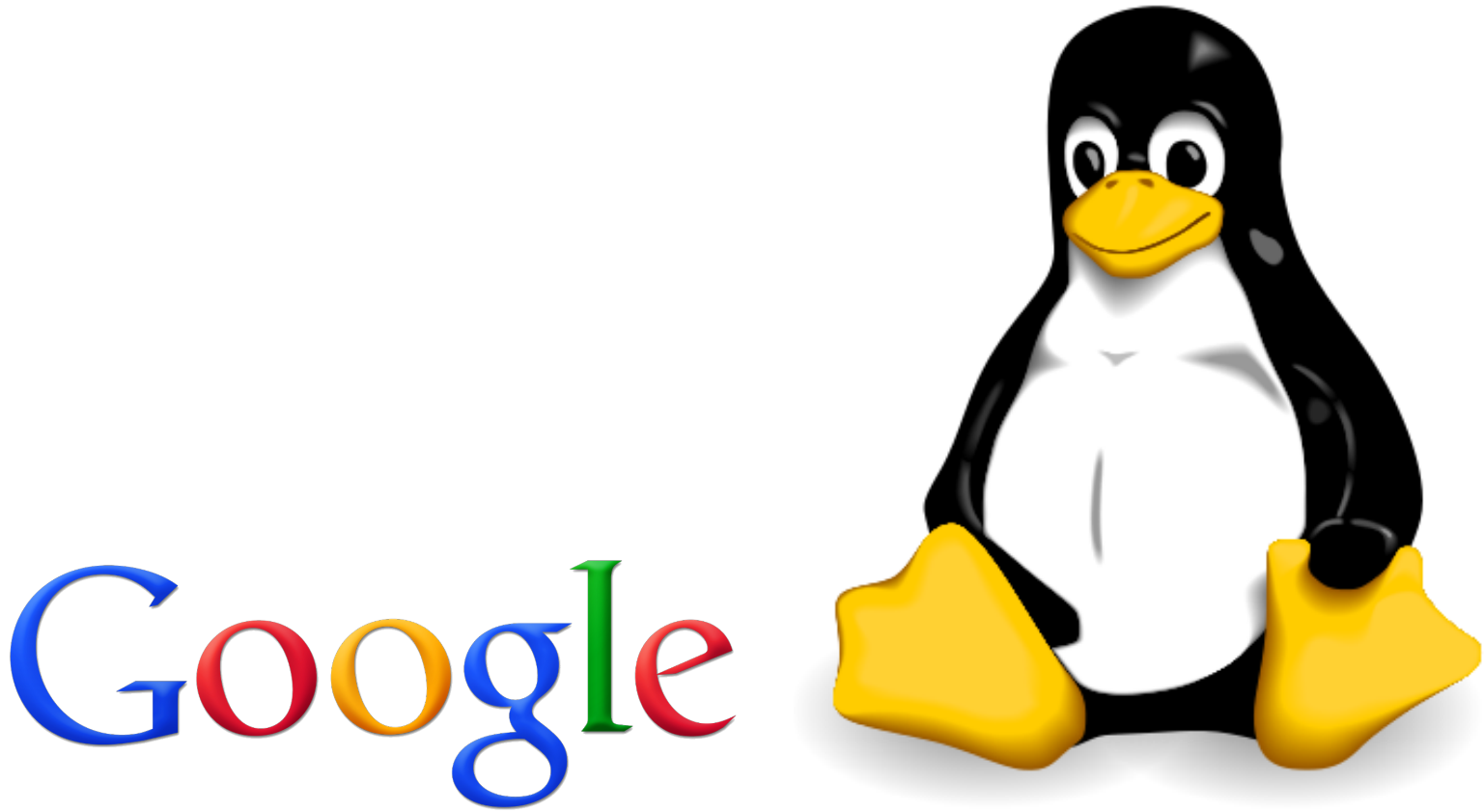
Super-reviews by “strong hackers”.

- understand the way Mozilla code is supposed to look,
- need not have domain expertise.

They look out for:

- proper use of APIs;
- adherence to Mozilla’s portability guidelines;
- cross-module effects; and
- respect of Mozilla coding practices.

Other organizations



Many organizations, including Google and Linux kernel hackers, review extensively.
Gerrit[2]: a tool out of Google for code reviews.

[2] http://lwn.net/Articles/359489/