CS 4500 Software Development

Code Reviews

Ferdinand Vesely

October 25, 2019

Usually:

- Requirements and design – meetings, discussions, critique
- Input from customers, managers, developers, and QA to synthesize a result.

Why not for code?

Books

Nothing is commercially published without scrutiny and input from editors

Code Reviews

- To discover defects in the design or code
- Part of the QA process, along with testing

Important:

Not to criticize the author, but to critique the code.

Benefits

Direct benefits:

- Improved code quality
- Fewer defects in code
    - Inspections typically catch 60% of defects
- Improved communication about code content
- Education of junior programmers

Types

- Formal inspections
- “Over-the-shoulder” reviews
- E-mail pass-around reviews
- Tool Assisted Reviews
- Instant Review (Pair Programming)

Formal Inspections

- Heavy-process review
- 3-6 participants
- Specific roles
- Formal process
- Traceable, measurable

Formal Inspections

Roles

1. Moderator / controller
   - Organizer (room, scheduling, distributing artifacts)
   - Keep everyone on task
   - Pace of review
   - Arbiter of disputes
2. Reviewer
   - Critical analysis
3. Reader
   - Looks at source code for comprehension
   - Presents this to the group
   - Author does not present the code to the group
   - This separates what the author intended from what is actually presented

Roles

4. Scribe
    - Record errors
    - Produce action items
5. Observer
    - E.g., domain-specific advice or learning
6. Author
    - Explain unclear parts of design or code
    - Occasionally: explain why things that seem like errors but are fine
    - Might present an initial overview of the project

Procedure

1. Planning
    - Author gives code to moderator
    - Moderator picks reviewer(s), time and place
    - Distributes code + checklist
2. Overview
    - If reviewers unfamiliar with project
    - By author – shouldn't speak for the code
    - Risky
3. Preparation
    - Reviewers scrutinize code individually
    - Different reviewers might have different perspectives or scenarios to check

Procedure

4. Meeting
    - Reader reads (paraphrases) the code
    - All logic is explained
    - Scribe records errors as they are discovered
    - Moderator moves discussion along, keeps it focused
    - Not too slow or too fast – around 150-200 nonblank, noncomment lines per hours is a good place to start
    - No discussion of solution – focus on discovering defects or shortcomings
    - Not more than 2 hours

Report

- Defects recorded in detail
- Location
- Severity
- Type

Report

Additionally, metrics are recorded:

- Individual time spent
- LOC inspection rates
- Process improvement

Pros / Cons

Pros

- Many people spending time reading code
- Potentially many defects identified
- “Paper trail”

Cons

- Ties up many people for a considerable amount of time
- Complex meeting preparations
- Training might be needed

Over-the-shoulder Reviews

- Most common informal review
- A developer (who did not participate in development) reviews while author walks through a set of code changes
- Author drives the review
- Resolution: “spot pair-programming” for small fixes
- Bigger changes taken off-line
- Remote alternative using screen-sharing software

Over-the-shoulder Reviews

- Simple to execute
- But: not an enforceable process
- Easy for author to miss changes after review is done
- Fixes for found bugs usually not verified
- +/- Author controls the pace of the review

Email Pass-around Reviews

- Whole files/changes packaged up and sent to reviewers via email
- Reviewers discuss, suggest changes
- Support for this in, e.g., Git: `git format-patch`
- Used by many open-source projects (Linux kernel, Git itself) – via mailing lists

Email Pass-around Reviews

- Easy to implement
- Can reach more people
- Easy to involve extra reviewers if needed
- Does not disrupt reviewers' work
- Can be difficult to track / follow the email conversation

Tool-assisted Reviews

- Software to assist with various aspects of review process
- Checklist & Workflow management
- Integrations with VC systems,
- Reports and metrics (process improvement)
- Audit management
- E.g., Smartbear Collaborator
- Lighter: Github pull requests

Pair Programming – Instant Reviews

- Reviewing developer is deeply involved in the code
- Better consideration for issues and consequences arising from different implementations
- Reviewer has more time and deeper insight
- But: reviewer cannot take a step back and review from a fresh & unbiased position

Ego

- Someone looking over your work
- Probably some attachment to it
- Criticisms: sometimes hard not to take personally
- Acknowledge a criticism and move on
  - Doesn’t imply that the author agrees with the content of the criticism
- Author should not try to defend the work under review

Checklists

- Common programming errors
- Based on examples in literature or experience
- Might be different for different implementation languages
- Might include coding guidelines

Fault class	Inspection check
Data faults	■ Are all program variables initialized before their values are used? ■ Have all constants been named? ■ Should the upper bound of arrays be equal to the size of the array or Size −1? ■ If character strings are used, is a delimiter explicitly assigned? ■ Is there any possibility of buffer overflow?
Control faults	■ For each conditional statement, is the condition correct? ■ Is each loop certain to terminate? ■ Are compound statements correctly bracketed? ■ In case statements, are all possible cases accounted for? ■ If a break is required after each case in case statements, has it been included?
Input/output faults	■ Are all input variables used? ■ Are all output variables assigned a value before they are output? ■ Can unexpected inputs cause corruption?
Interface faults	■ Do all function and method calls have the correct number of parameters? ■ Do formal and actual parameter types match? ■ Are the parameters in the right order? ■ If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	■ If a linked structure is modified, have all links been correctly reassigned? ■ If dynamic storage is used, has space been allocated correctly? ■ Is space explicitly de-allocated after it is no longer required?
Exception management faults	■ Have all possible error conditions been taken into account?

Summary

Code reviews:

- A reviewer goes through code, looking for defects shortcomings
- Can be informal, or formal with predefined deliverables
- Integration with VCS, also standalone tools
- Effective technique
- Low requirements (informal)