Working in Teams

Chung-Kil Hur

(Credit: Byung-Gon Chun & Many Slides from UCB CS169 taught by Armando Fox and David Patterson)

Outline

- Design Reviews and Code Reviews
- The Five R’s of Bug Fixing
- Pitfalls

Agile Design and Code Reviews??

Good Meetings: SAMOSAS

(Photo by K.S. Poddar. Used by permission under CC-BY-SA-2.0.)

- Start and stop meeting promptly
- Agenda created in advance; no agenda, no meeting
- Minutes recorded so everyone can recall results
- One speaker at a time; no interrupting talker
- Send material in advance, since reading is faster
- Action items at end of meeting, so know what each should do as a result of the meeting
- Set the date and time of the next meeting

Minutes and action items record results of meeting, start next meeting by reviewing action items

Design/Code Reviews

- Design review: meeting where authors present design with goal of quality by benefiting from the experience of the people attending the meeting
- Code review: held once the design has been implemented
- In the Agile/Scrum context, since design and implementation occur together, they might be held every few iterations

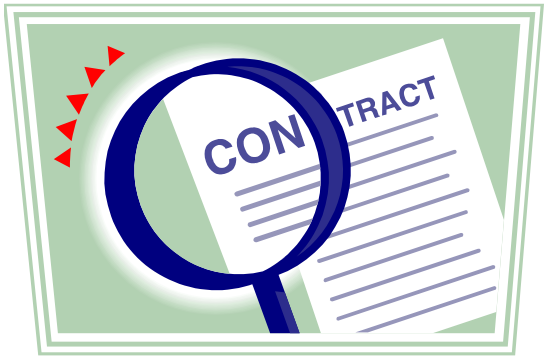
Design/Code Reviews

- Shalloway*: formal design and code reviews often too late in process to make big impact
- Recommends instead have earlier, smaller meetings: "approach reviews".
  - A few senior developers assist team in coming up with an approach to solve the problem
  - Group brainstorms about different approaches
- If do a formal design review, suggests 1st hold a "mini-design review" to prepare

*Alan Shalloway, *Agile Design and Code Reviews*, 2002, www.netobjectives.com/download/designreviews.pdf

Code Reviews
Can Check Comments too

- Challenge: keeping comments consistent with changes, refactoring



- Code review is one place to ensure comments make sense

Comment Example

```
# Add one to i.
i += 1

# Lock to protect against concurrent access.
mutex = SpinLock.new

# This method swaps the panels.
def swap_panels(panel_1, panel_2)
  # ...
end
```

Comment Example

```
# Loop through every array index, get the
# third value of the list in the content to
# determine if it has the symbol we are
looking
# for. Set the result to the symbol if we
# find it.

# Good Comment:
# Scan the array to see if the symbol exists
```

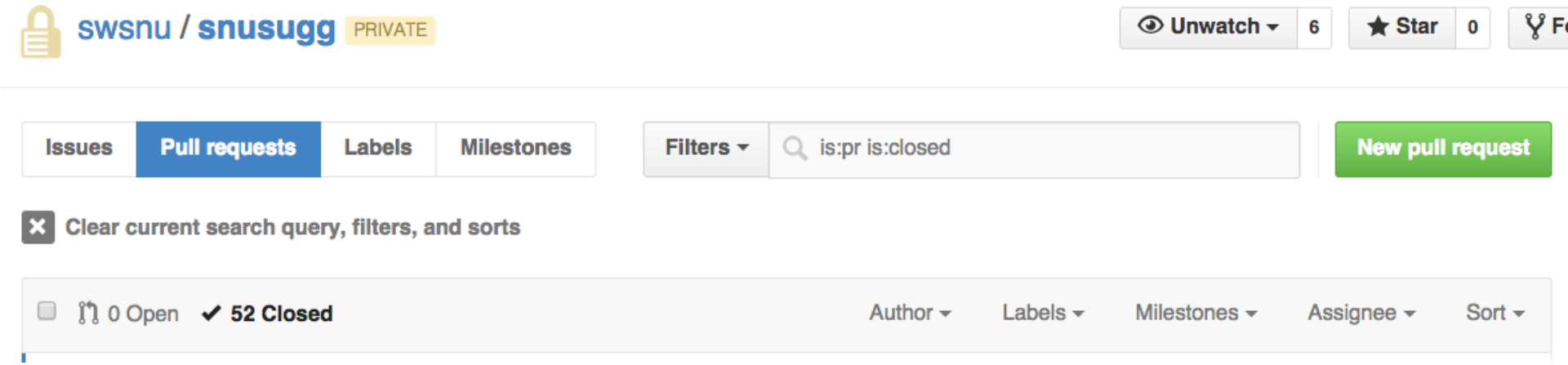
Advice on Comments

- Document **what is not obvious** from code
  - Don't just repeat what's obvious from the code
- **Raise level of abstraction**
  - Do think about what's not obvious at both the low level and the high level
- Explain **why** did something
  - Design issues that went through your mind while you were writing the code

Reviews and Agile?

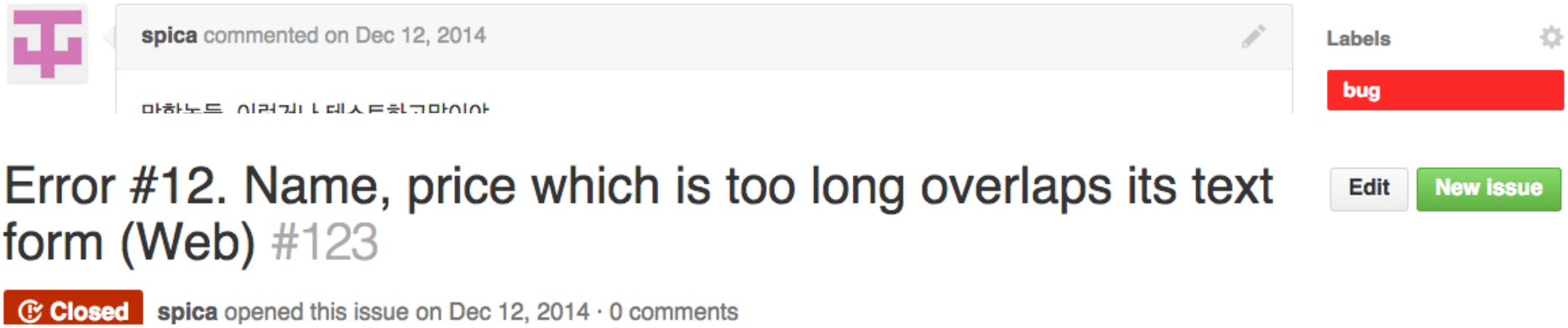
- Pivotal Labs – pair programming makes review superfluous
- GitHub – *pull requests* replace code reviews
  - 1 developer requests her code to be integrated with code base
  - Rest of team see request and determines how affect their code
  - Any concern leads to online discussion
  - Many **pull requests**/day
    => many **minireviews**/day

Code Review Example (Good)



https://github.com/swsnu/snusugg/pull/49

Code Review Example (Bad)



Which expression statement regarding Design Reviews and Meetings is FALSE?

- ☐ Intended to improve the quality of the software product using the wisdom of the attendees
- ☐ They result in technical information exchange and can be highly educational for junior people
- ☐ Design reviews can be beneficial to both presenters *and* attendees
- ☐ Serving food like Samosas is vital to success of a good meeting

Fixing Bugs: The Five R’s

No Bug Fix Without a Test!

- **R**eport
- **R**eproduce and/or **R**eclassify
- **R**egression test
- **R**epair
- **R**elease the fix (commit and/or deploy)

- Even in non-agile organizations
- But, existing agile processes can be *adapted* to bug fixes

Report

- GitHub “issues” feature
- Full-featured bug tracking, eg Bugzilla,JIRA
- *Use the simplest tool that works for your team & project scope*

Reclassify?
or Reproduce + Repair?

- Reclassify as “not a bug” or “won’t be fixed”
- Reproduce with *simplest possible* test, and add it to regression
  - minimize preconditions

  No bug can be closed without a test.
- Release: may mean either “pushed” or “deployed”

Which type of bug could you *not* create a regression test for?

- Bug that depends on behavior of external service
- Bug that depends on time of day, day of year, etc.
- Bug that is statistical in nature ("Player must win jackpot at least X% of the time")
- You can reproduce any of these dependencies in regression tests

Pitfall

- Pitfall: Dividing work based on the software stack rather than on features
- E.g., Front-end specialist, back-end specialist, customer-liaison, …
- Better for each team member to deliver all aspects of a chosen story
- Better for app, better for career growth

Summary

- Version Control : Git
- Design and Code Reviews
- Fixing Bugs: The Five Rs
- When project done, take time to think about what learned before leaping into next one
  - What went well, what didn't, what do differently