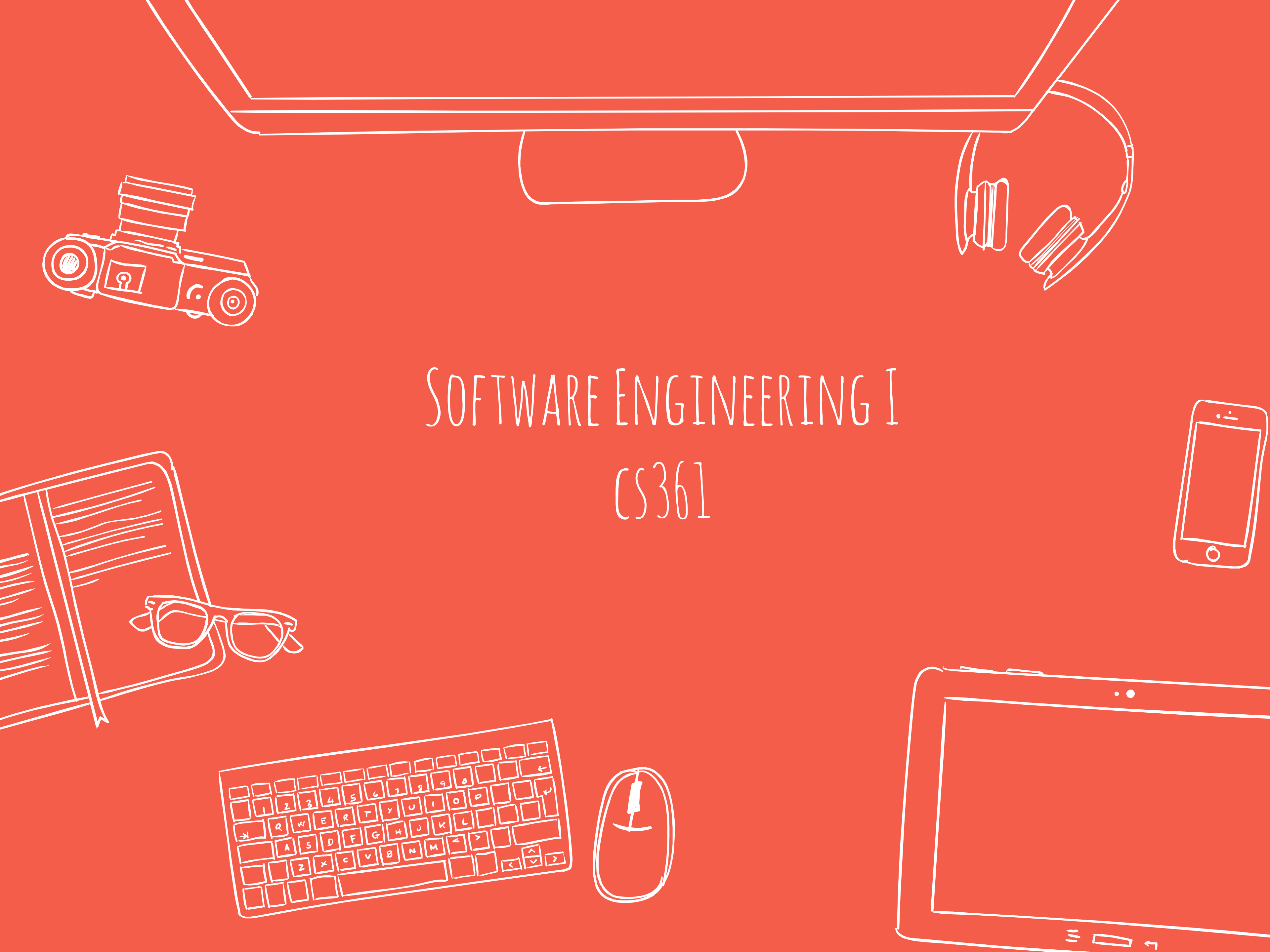


Software Engineering I

CS361

Announcements

- ✖ GitKraken gitkraken.com
- ✖ Writing Assignment 3 Feedback

Code Reviews

Showing off code for the sake of common interest

- ✖ http://fabiensanglard.net/prince_of_persia/index.php

ATTRIBUTION

Much of this material inspired by a great slides from Adam Badura, available here: https://www.linkedin.com/pulse/my-lecture-code-review-from-codedive-2015-conference-adam-badura

“Code review is having other people look at your code in order to find defects.”

Code Review Pros and Cons

+ prevents releasing bugs
+ ensures architecture quality
+ leads to personal development

– takes time
– is impractical when reviewer doesn’t know domain
– hurts feelings

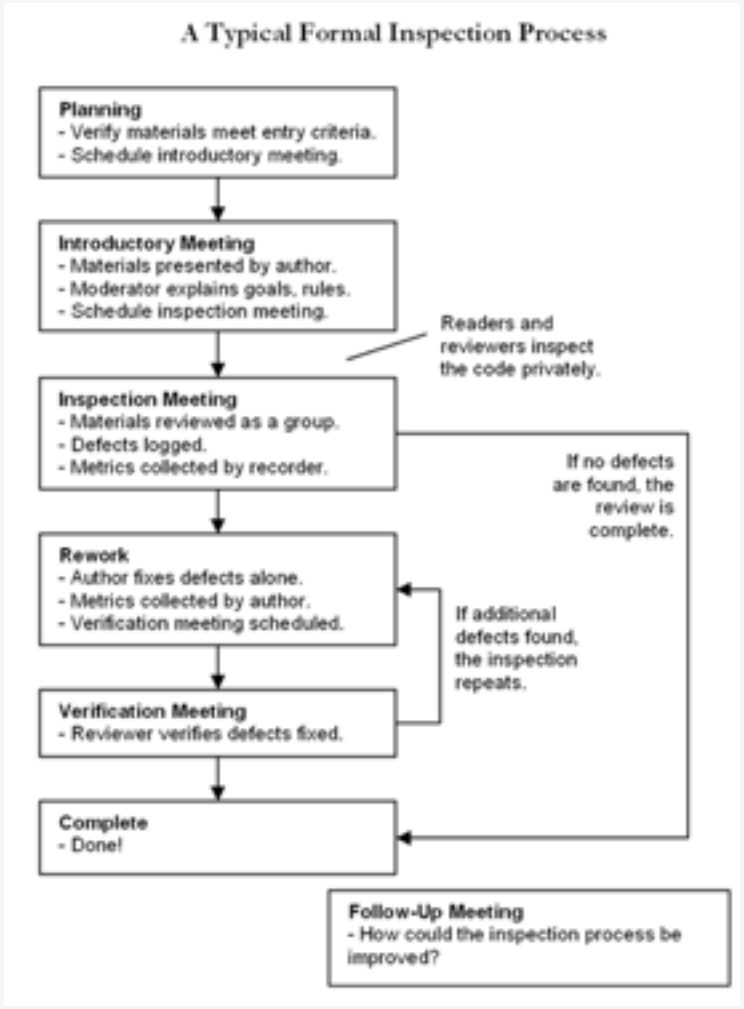
Formal Inspection

✖ First developed by Michael Fagan in the mid 1970’s.

✖ Very Specific Heavyweight process with 4 roles and 7 steps

Not Michael Fagan who broke into the Queens bedroom

Formal Inspection


A Typical Formal Inspection Process
Planning
- Verify materials meet entry criteria.
- Schedule introductory meeting.
Introductory Meeting
- Materials presented by author.
- Moderator explains goals, rules.
- Schedule inspection meeting.
Readers and reviewers inspect the code privately.
Inspection Meeting
- Materials reviewed as a group.
- Defects logged.
- Metrics collected by recorder.
If no defects are found, the review is complete.
Rework
- Author fixes defects alone.
- Metrics collected by author.
- Verification meeting scheduled.
If additional defects found, the inspection repeats.
Verification Meeting
- Reviewer verifies defects fixed.
Complete
- Done!
Follow-Up Meeting
- How could the inspection process be improved?

http://www.methodsandtools.com/archive/archive.php?id=66

Formal Inspection

- ✖ It Works, but is expensive.
- ✖ 9 person-hours per 200 lines of code
- ✖ Very impractical for today’s realities

Lighter weight approaches

- Over the Shoulder
- Pair Programming
- Pull Requests

Over the shoulder

✖ Reviewer sits with the developer and looks “over their shoulder” at the code.

✖The reviewer can give informal feedback which can then be incorporated immediately if possible

Over the Shoulder

+ Easy to Implement
+ Fast to Complete
+ Easy to quickly incorporate changes
– Reviewer cannot review at their own pace
– No Verification
– Reviewer only sees that developer shows them

Pair Programming

- ✖ Code is written by a pair, so Code Review is “Baked In” to the process.
- ✖ We will discuss later today

Pair Programming

+ Great for finding bugs and promoting knowledge transfer
+ Review is in-depth
- Reviewer is not objective
- Hard to do remotely
- No Verification

Pull Requests

- ✖ Code is peer reviewed as a part of the Pull Request process
- ✖ No pull request should be accepted without being reviewed by a different developer

Pull Request Code Reviews

+ Can be enforced by Version Control Practices
+ PR serves as verification of review
+ Can be done asynchronously
+ Reviews can see all source code
- Might be hard to understand without explanation
- Most important changes can be lost with lots of small insignificant changes

Peer Review Best Practices: Architecture/Design

Single Responsibility Principle
Code Duplication
Squint Test
Left Code Better
Potential Bugs
Error Handling
Efficiency

http://kevinlondon.com/2015/05/05/code-review-best-practices.html

Peer Review Best Practices: Style

- Method Names
- Variable Names
- Function Length
- Class Length
- File Length
- Commented Code
- Number of Method Arguments
- Readability

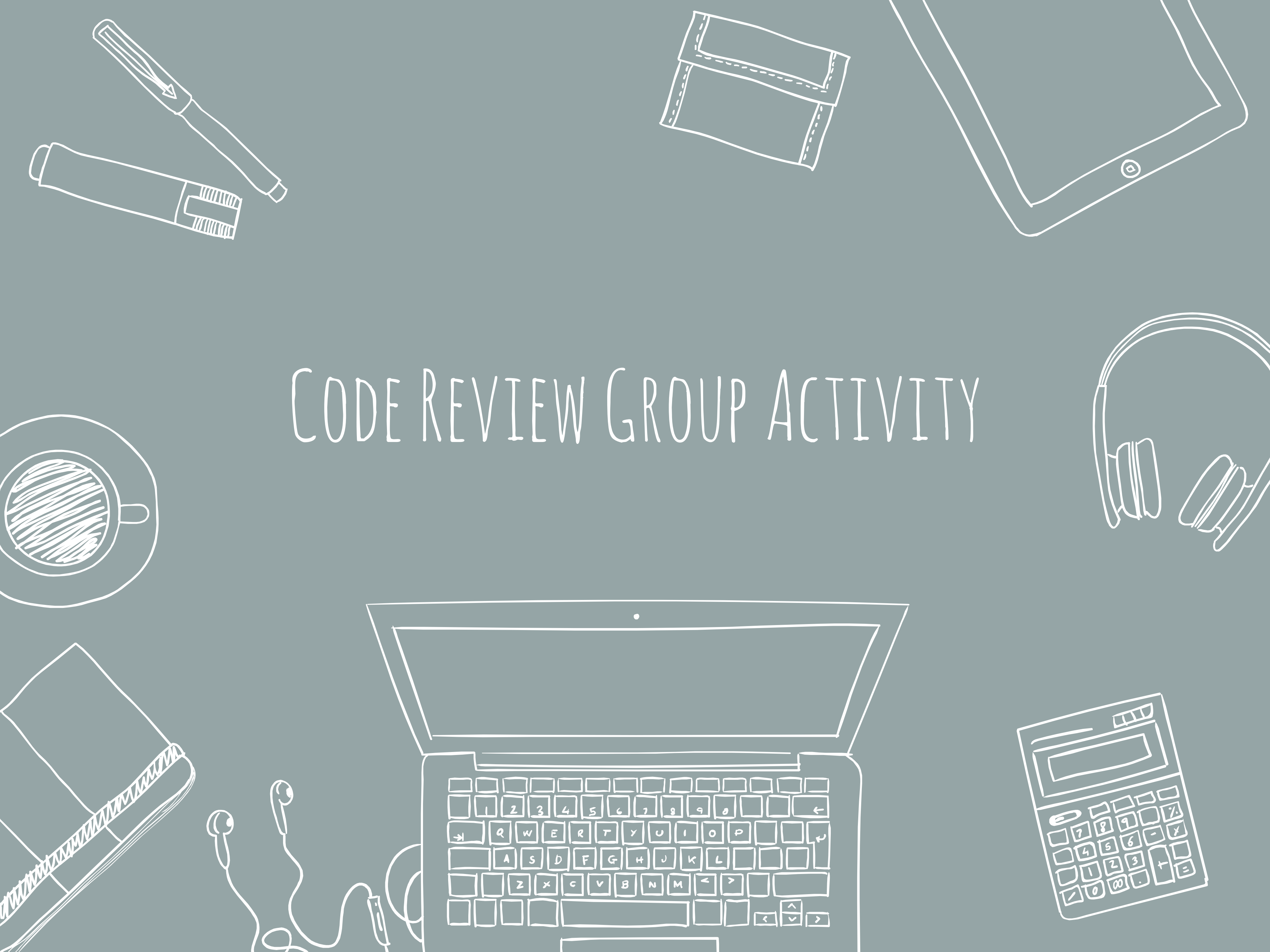
Peer Review Best Practices: Testing

- ✖ Test Coverage
- ✖ Testing at the right level
- ✖ Number Mocks
- ✖ Meets requirements

Practical Suggestions

- ✖ Review < 400 LOC at a time
- ✖ Don't review > 60 min at a time
- ✖ Use a Peer Review Checklist (should be domain/language specific)
- ✖ Follow up with review comments

https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/

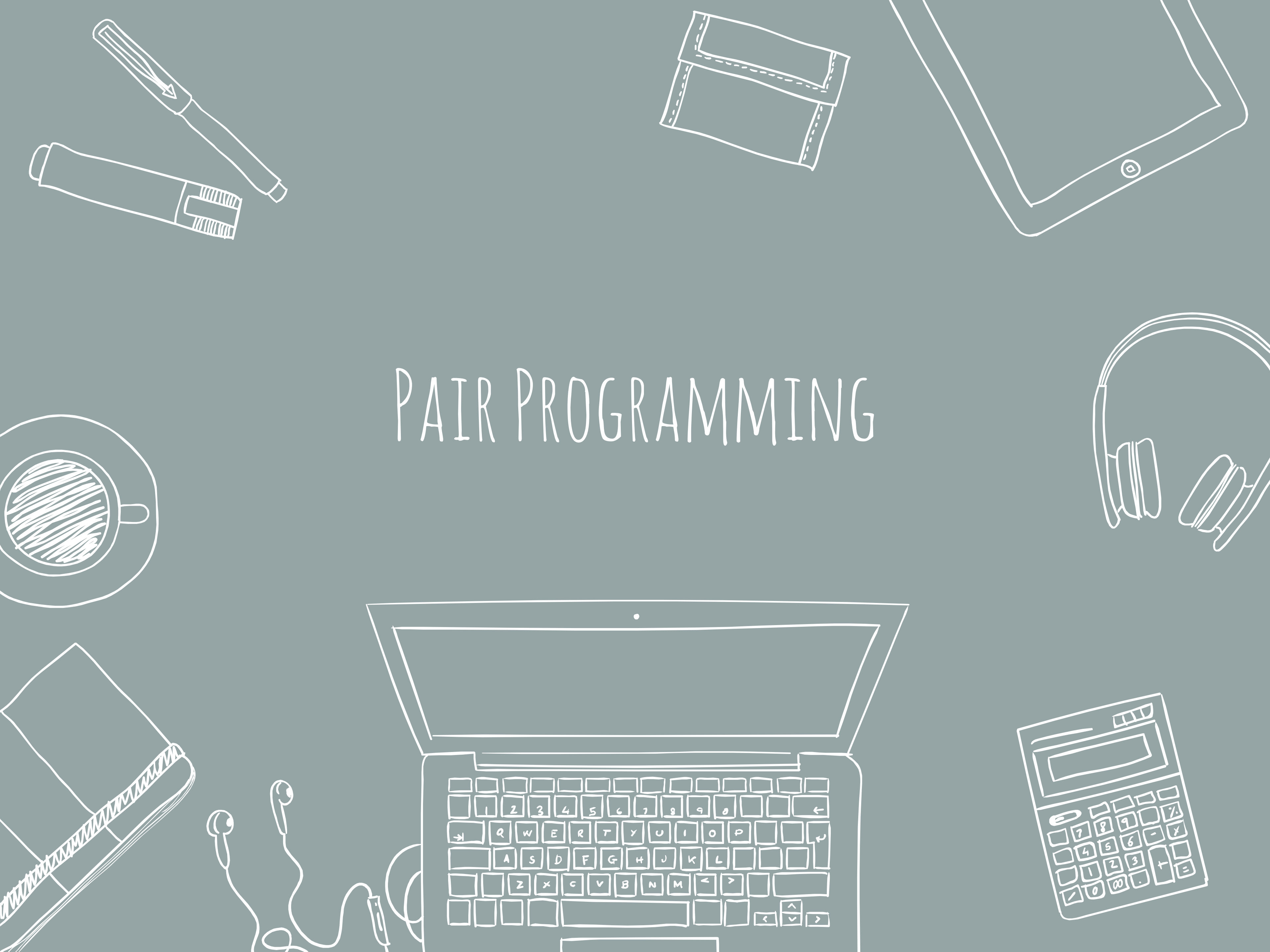


Code Review Group Activity

Tools to help

- ✖https://www.codereviewhub.com/
- ✖https://www.jetbrains.com/upsource/
- ✖https://www.reviewboard.org/
- ✖https://reviewable.io/
- ✖https://www.gitcolony.com/
- ✖https://www.review.ninja/

Pair Programming



XP Practices

- Pair Programming
- TDD
- Continuous Integration
- Refactoring
- Small Releases
- Coding Standards
- Collective Code Ownership
- Simple Design
- Sustainable Pace

XP Practices

- **Pair Programming**
- TDD
- Continuous Integration
- Refactoring
- Small Releases
- Coding Standards
- Collective Code Ownership
- Simple Design
- Sustainable Pace

Pair Programming

- ✖ 2 Programmers, single computer
- ✖ Driver:
  Controls the mouse/keyboard
  Deals with the details
- ✖Navigator:
  Thinks at a higher level
  Watches for typos, logical errors
- ✖Switch off every 10-20 minutes

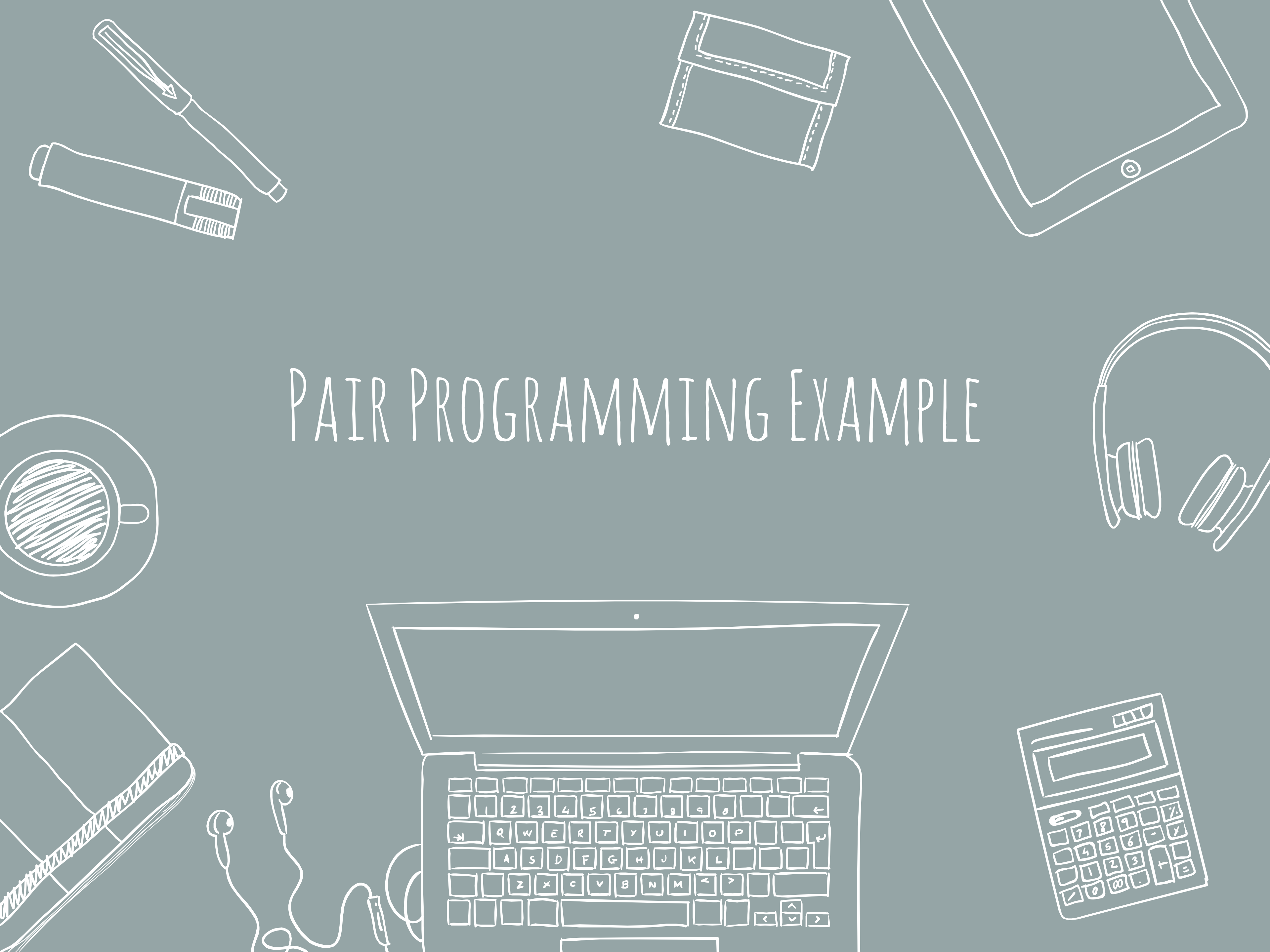
Why Pair Program?

- ✖ Leads to less defects
- ✖ Leads to higher design quality
- ✖ Higher programmer job satisfaction
- ✖ Knowledge is shared for continuous learning
- ✖ Team-building and communication is enhanced
- ✖ Raises your team’s bus number

Why not to pair program

- ✖ Two people cannot be physically present
- ✖ Strong personality conflicts
- ✖ When the task is simple and unchallenging
- ✖ When participants need a break

Pair Programming Example



Credits

Special thanks to all the people who made and released these awesome resources for free:

- ✖ Presentation template by SlidesCarnival
- ✖ Photographs by Unsplash