Design and Code Reviews - Overview
What are design and code reviews?

Why should you do them?

The PSP
• review strategy
• review principles
• review measures

Review considerations
What are Reviews?

A review is a way to personally examine your own work.

It is one of a family of methods
• inspections
• walkthroughs
• reviews

They all have the objective of finding and fixing product defects early in the development process.

Inspections

Inspections were introduced by Fagan at IBM in 1976.

Inspections follow a structured process
• done by peers
• managers do not attend
• each participant has a defined role
• preparation is required
• objective is to find problems

Inspections are useful for requirements, designs, code, test cases, plans, etc.
Walkthroughs

Walkthroughs typically follow a meeting format
• developer leads the audience through the product
• audience may include peers, managers, or other interested parties
• objective is to communicate or obtain approval
• no preparation is required

Walkthroughs are most useful for requirements and system design issues.

Reviews

In a personal review
• professional privately reviews his/her product
• objective is to find defects before the first compile and test
• reviews are most effective when structured and measured

Reviews can be used for requirements, designs, and code.
Why Do Reviews? - 1

Data show that it is much more efficient to find defects in reviews than in testing
- in unit test, typically only about 2 to 4 defects are found per hour
- code reviews typically find about 10 defects per hour
- experienced reviewers can find 70% or more of the defects in a product
- unit test rarely exceeds a 50% yield

PSP data show that reviews find 2 to 5 times as many defects per hour as unit test
Why Do Reviews? - 2

After unit test, defect removal becomes much more expensive
  • in integration and system test it takes 10 to 40 programmer hours to find and fix each defect
  • inspections typically take less than an hour per defect

Some inspection data
  • O’Neil: 80-90% yield at 25 minutes/defect
  • Philips: 15 minutes/defect versus 8 hours in test

Why Do Reviews? - 3

The reason to eliminate defects as early as possible is because
  • every review, inspection, compile, and test finds only a fraction of the defects
  • thus, the cleaner the code entering a phase, the cleaner it will be on exit.

Early defect removal saves time and money
  • defects cost more to find and fix later in the development process
  • defects cost much more to find and fix after development completion
Why Reviews are Efficient - 1

In testing
• you start with a problem
• then you must find the bug
• next, you devise a fix
• finally, you implement and test the fix

With reviews and inspections
• you see the defect
• then you devise a fix
• finally, you implement and review the fix

Why Reviews are Efficient - 2

In testing, it the system produces an unusual result, then you must
• detect that it was unusual
• figure out what the test program was doing
• find where it was in your program
• figure out what defect could cause such behavior

You are searching for the unplanned and unexpected.

This can take a lot of time
Why Reviews are Efficient - 3

With reviews and inspections
• you follow your own logic
• when you find a defect, you know exactly where you are
• you know what the program should do and did not do
• you thus know why this is a defect
• you are therefore in a better position to devise a complete and effective fix

The PSP Review Strategy

The PSP objective is to produce the highest possible program quality at every process phase

The review strategy to achieve this is to
• gather data on your reviews
• study these data
• decide what works best for you
• adjust your process accordingly

This is a continuous learning process using data on your own work.
Review Principles

PSP reviews follow a disciplined process with
• entry and exit criteria
• a review structure
• guidelines, checklists, and standards

The suggested PSP review goal is to find every defect before the first compile or test.

To address this goal, you should
• use coding standards
• use design completeness criteria
• measure and improve your review process

Design Review Principles

Produce designs that can be reviewed.

Follow an explicit review strategy.

Review the design in stages.

Verify that the logic correctly implements the requirements.
Produce Designs that can be Reviewed

A reviewable design has
- a defined context
- a precise representation
- a consistent and clear structure

This suggests that
- the design’s purpose and function be explicitly stated
- you have criteria for design completeness
- the design is structured in logical elements

Follow a Review Strategy

The review strategy specifies the order in which you review the design elements.
- this depends on the product structure
- the review strategy should thus be considered during design

The objective should be to produce a design that can be
- reviewed in stages
- tested in stages
- integrated in stages
Review the Design in Stages

By reviewing in stages, you ensure that all selected topics are carefully checked.
Suggested review stages are
1. check that all elements have been designed
2. verify overall program structure and flow
3. check the logical constructs for correctness
4. check for robustness
5. check the function, method, and procedure calls to ensure proper use
6. check special variables, parameters, types, and files for proper use

Verify that the Logic Correctly Implements the Requirements

Review the requirements to ensure that each required function is addressed by the design.

Check carefully for oversights and omissions.
Review Measures

Explicit measures
• the size of the program being reviewed
• the review time
• the numbers of defects found
• the numbers of defects not found: the escapes

Derived measures
• review yield: %found
• LOC reviewed per hour
• defects found per KLOC
• defects found per review hour
• review leverage

Review Yield - 1

Yield
• a measure of process quality
• the percent of defects in the product at review time that were found by the review
• measures the effectiveness of a process step
  - design and code reviews
  - the overall process - prior to test
  - the development process - including test

yield(for a phase or the entire process) =
100*(defects found)/(defects found + not found)
Review Yield - 2

Yield cannot be calculated until all defects have been found through test and product use.

Yield can be useful early in the process if all or most defects are counted.
  • design and code review defects
  • compile defects
  • unit test defects

Using process data, control parameters can help to ensure high yield reviews.

Defect Removal Leverage: DRL

DRL measures the relative effectiveness of a process step at removing defects.

DRL is the number of defects removed per hour by a process step relative to a base process
  • the usual base is unit test
  • DRL(X/UT) is the DRL for phase X with respect to unit test

DRL is calculated as follows: DRL(X/UT) = (defects/hour phase X)/(defects/hour unit test)
Process Control

To control your process, you need measures that are available while you are enacting the process.

While yield and DRL are very helpful, they are not available until after process completion.

The need is for measures that
• correlate with yield
• are available at development time

Potential Control Parameters

The potential control parameters for yield are
• LOC reviewed per hour
• defects found per hour
• defects found per KLOC

The PSP research has found the following correlations with yield
• LOC/Hour: -0.63, with significance > 0.005
• Defects/Hour: 0.14, with significance of 0.25
• Defects/KLOC: 0.47, with significance of 0.01

While none are good, LOC/Hour is best.
Yield vs. LOC/Hour - Student 12

Yield vs LOC/Hour - Student 20
LOC Reviewed per Hour

Student data has considerable variation.

These examples show that structured reviews can be effective if
• they follow a defined procedure
• they use checklists
• they are tailored to each engineer’s defect experience

For the PSP, a suggested control target is to review no more than about 200 LOC per hour.

Review Considerations

Checklists

Reviewing before or after compile

The relationship of reviews and inspections
Checklists

When performing precise tasks, it is difficult to do more than one thing well at a time.

The checklist defines the review steps in the order suggested for performing them.

By checking off each item, you are more likely to perform it properly.

Establish a personal checklist that is customized to your defect experience.

Reviewing Before Compile - 1

The PSP calls for code reviews before the first compile.

The reasons are
• review time is about the same whether done before or after compile
• code reviews will find syntax defects that the compiler misses
• code reviews of compiled code tend to be much less effective
• the compiler is equally effective before or after reviews
Reviewing Before Compile - 2

The PSP uses compile as a check on the quality of the reviews.
  • if too many defects are found in compile, another review is called for
  • if the compile is clean, it is likely that most of the defects have been found

After you have completed the PSP exercises, you may wish to gather data on your reviews both before and after compile and see which are most effective for you.

Reviews and Inspections

The principal focus of inspections should be to find those requirements and design problems that you have missed.

When programs have many simple defects, inspectors are distracted and often miss more important problems.

Reviewing the code first
  • provides a quality product for the inspection
  • shows respect for the inspectors’ time
  • produces higher quality inspections
Assignment #7

Read chapter 8 of the text and produce report R4. Check Appendix D for the report requirements.

This assignment calls for you to
• design a process for producing the R4 report
• include a planning and postmortem phase in this process
• plan the R4 report task
• produce the report
• submit the report, the process, and the process data

Messages to Remember from Lecture 7

Design and code reviews effectively
• improve the quality of your programs
• save development time

To do effective reviews, you must
• establish review goals
• follow a disciplined review process
• measure and improve your review quality