

CS 350: Introduction to Software Engineering

Slide Set 5
Software Quality
C. M. Overstreet
Old Dominion University
Spring 2006

Lecture Topics

- What is quality?
- The economics of quality
- Defect-removal methods
- Design and code reviews
- Quality measures
- Review considerations

What is Quality?

- Basic definition: meeting the users' needs
 - *needs*, not wants
 - true functional needs are often unknowable
- There is a hierarchy of needs that differ depending on needs
 - Do the required tasks.
 - Meet performance requirements.
 - Be usable and convenient.
 - Be economical and timely.
 - Be dependable and reliable.

The PSP Quality Focus -1

- To be useful, software must
 - install quickly and easily
 - run consistently
 - properly handle normal and abnormal cases
 - not do destructive or unexpected things
 - be essentially bug-free
- Defects are not important to users, as long as they do not
 - affect operations
 - cause inconvenience
 - cost time or money
 - cause loss of confidence in the program's results

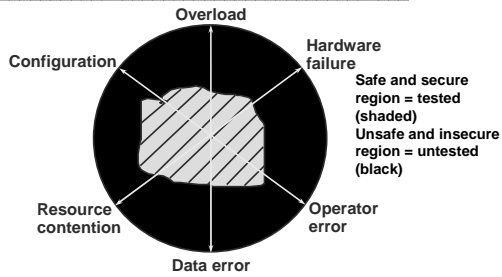
The PSP Quality Focus -2

- The defect content of software products must be managed before more important quality issues can be addressed.
- Low defect content is an essential prerequisite to a quality software process.
- Since low defect content can best be achieved where the defects are injected, engineers should
 - remove their own defects
 - determine the causes of their defects
 - learn to prevent those defects

The Economics of Quality

- Software is the only modern technology that relies on testing to manage quality.
- With common quality practices, software groups typically
 - spend 50+% of the schedule in test
 - devote more than half their resources to fixing defects
 - cannot predict when they will finish
 - deliver poor-quality and over-cost products
- To manage cost and schedule, you must manage quality.
- To get a quality product out of test, you must put a quality product into test.

Testing Alone is Ineffective



Fall 2005

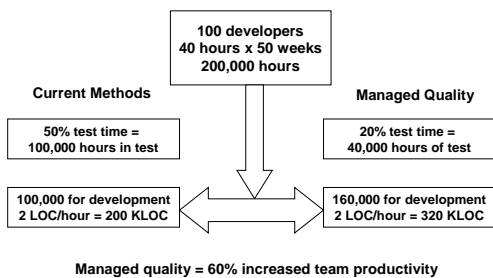
CS 350/ODU

7

Removing Defects in Test

- When performing a task thousands of times, economics would suggest that you use the most efficient methods.
- A 50,000 LOC system with traditional development methods would
 - have 25+ defects/KLOC at test entry - 1250 defects
 - take 12,500+ programmer hours to test
 - be late and over budget
- At the typical rate of 10+ hours per defect, this is 6 programmer years.

Quality and Productivity

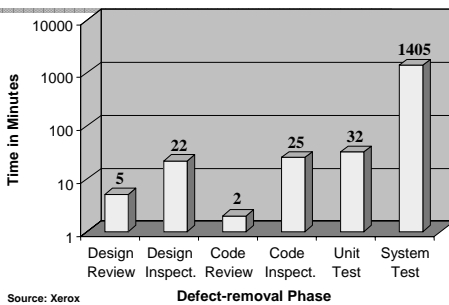


9

Defect-removal Methods -1

- The principal ways to find and fix defects are by
 - compiling
 - unit testing
 - integration and system testing
 - team inspections
 - personal reviews
- Since you will likely have to remove lots of defects, you should use the most efficient methods.

Defect-removal Times



Defect-removal Methods -2

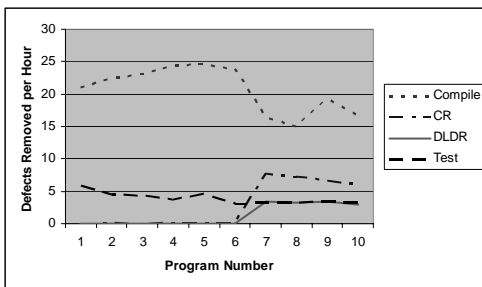
- In a personal review
 - you privately review your product
 - your objective is to find and fix defects before test
- Reviews are most effective when they are structured and measured.
- Use reviews for requirements, test plans, code design, code, test data, and everything else that you develop.
- Also continue to use inspections, compiling, and testing.

Defect-removal Rates -1

- Even at the personal level, it is more efficient to find defects in reviews than in testing.
 - Unit test finds only about 2 to 4 defects per hour.
 - Unit test finds about 50% of the defects.
 - Code reviews find about 6 to 10 defects per hour.
 - Practiced reviewers can find 70% or more of the defects before compiling or testing.

Defect-removal Rates -2

810 Developers



Why Reviews are Efficient

- In testing, you must
 - detect unusual behavior
 - figure out what the test program was doing
 - find where the problem is in the program
 - figure out which defect could cause such behavior
- This can take a lot of time.
- With reviews and inspections, you
 - follow your own logic
 - know where you are when you find a defect
 - know what the program should do, but did not
 - know why this is a defect
 - are in a better position to devise a correct fix

Review Principles

- PSP reviews follow a defined process with guidelines, checklists, and standards.
- The PSP review goal is to find every defect before the first compile or test.
- To address this goal, you should
 - review before compiling or testing
 - use coding standards
 - use design completeness criteria
 - measure and improve your review process
 - use a customized personal checklist

The Code Review Checklist

- Your reviews will be most effective when your personal checklist is customized to your own defect experience.
 - Use your own data to select the checklist items.
 - Gather and analyze data on the reviews.
 - Adjust the checklist with experience.
- Do the reviews on a printed listing, not on the screen.
- The checklist defines the review steps and the suggested order for performing them.
- Review for one checklist item at a time.
- Check off each item as you complete it.

Design Review Principles

- In addition to reviewing code, you should also review your designs.
- This requires that you
 - produce designs that can be reviewed
 - Two ways to survive a design inspection:
 - Produce a design that is so complex no one can understand it
 - Produce a design that is so simple everyone can understand it
 - follow an explicit review strategy
 - review the design in stages
 - verify that the logic correctly implements the requirements

Reviewable Designs

- A reviewable design has a
 - defined context
 - precise representation
 - consistent and clear structure
- This suggests that
 - the design's purpose and function be explicitly stated
 - you have criteria for design completeness
 - typical completeness criterion: be able to pass design to programmers
 - (sometimes other documents and refs may be required)
 - the design is structured in logical elements

The Design Review Strategy

- Produce designs that can be reviewed in stages.
- The suggested review stages are as follows.
 1. Review against the requirements to ensure that each required function is addressed by the design.
 2. Verify the overall program structure and flow.
 3. Check the logical constructs for correctness.
 4. Check for robustness, safety, and security.
 5. Check the function, method, and procedure calls to ensure proper use.
 6. Check special variables, parameters, types, and files for proper use.

One design review criteria:

- All requirements are directly addressed by a specific component in the design
 - All requirements can be tied to specific code
- Every component of the design is justified by a specific requirement statement
 - All code can be tied to a specific requirement
- Does this mean all requirements are met?
- Does this mean all code is needed?
- Ans.: No, but it can uncover common oversights.

Quality Measures

- To do efficient reviews, you must have measures.
- The PSP has many useful quality and process-control measures.
 - yield
 - review rate
 - defects found per unit of size
 - defects injected and removed per hour
 - defect-removal leverage
 - cost of quality (COQ)

Phase Yield

- Phase yield measures the
 - percentage of the defects in the product that were found by that phase
 - defect-removal effectiveness of that process step
- Yield can be used to measure the effectiveness of design and code reviews, inspections, compiling, and testing.
- Yield (for a phase) = $100 * (\text{defects found}) / (\text{defects found} + \text{not found})$

Yield Estimates

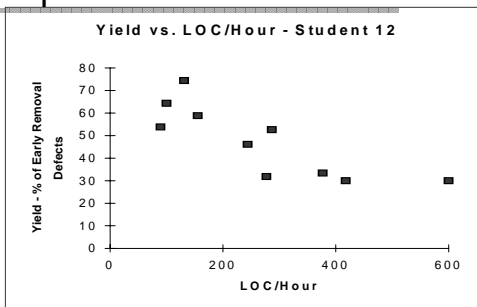
- Yield can be estimated but not precisely calculated until all defects have been found through test and product use.
- Yield measures are most useful when the developers and testers record all of the defects.
 - design and code review defects
 - compile defects
 - test defects
- By using process-control measures, you are more likely to do high-yield reviews.

Potential Control Parameters

- To be useful, process control measures must be available during the process. Examples are
 - size units reviewed per hour
 - defects found per hour
 - defects found per size unit
- While no control parameter directly correlates with phase yield, review rate is the most useful control parameter.
- Review rate is the parameter used in the PSP.

Yield versus Review Rate

-1



Yield versus Review Rate -3

- While there is considerable variation, higher rates generally give lower-yield reviews.
- The PSP suggests the following upper limits for review rates:
 - code (using the LOC measure): 200 LOC/hour
 - documents: 4 pages/hour
 - other measures: develop from your personal review data

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 (UT).
 - $DRL(X/UT)$ is the DRL for phase X with respect to unit test.
- DRL is calculated as:
$$DRL(X/UT) = \frac{\text{defects/hour phase X}}{\text{defects/hour unit test}}$$

Cost of Quality (COQ) -1

- COQ measure process quality in a way that is meaningful to management.
- The COQ elements are failure, appraisal, and prevention costs.
- Failure cost is the time spent in repair and rework plus the cost of any scrap. In PSP, it is compile and test time.
- Appraisal costs are the costs of inspecting for defects. In PSP, appraisal cost is design and code review time.
- In the TSP, inspections are included in appraisal costs.

Cost of Quality (COQ) -2

- Defect-prevention cost is the cost of identifying and resolving defect causes.
- Defect prevention generally is done by a team or a group and usually is led by a support staff member.
- Other defect-prevention costs are
 - formal specification and design work
 - prototyping
 - process analysis and improvement
 - defect recording

Cost of Quality (COQ) -3

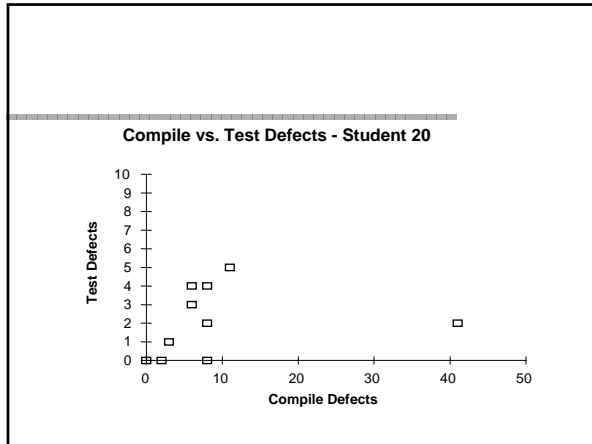
- A useful COQ measure is the ratio of appraisal to failure costs (A/FR). This is $A/FR = (\text{appraisal COQ}) / (\text{failure COQ})$
- A/FR experience
 - A/FR measures are sensitive to the project.
 - For the PSP exercises, the A/FR target is 2.0 or greater.
 - High A/FR is associated with low numbers of test defects and high product quality.
 - Projects should set A/FR targets based on estimated defect-free test times.

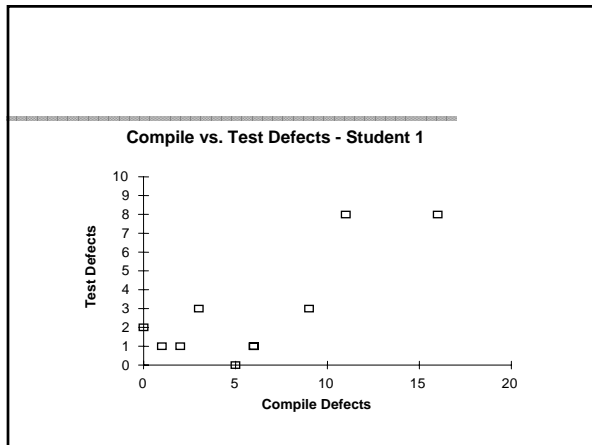
Review Considerations

- Reviewing before or after compile
- The relationship of reviews and inspections
- Defect prevention

Reviewing Before Compile

- When your development environment has a compile step, it is more efficient to do code reviews before compiling.
- The reasons for this are as follows
 - Review time is about the same before or after compiling.
 - If reviews are done before compiling, compile time is substantially reduced.
 - The time saved in compiling and testing is often more than the review time.
 - Code reviews find syntax/typo defects that the compiler would miss.
 - Code reviews of compiled code are less effective.
 - The compiler is equally effective before or after a review.
 - Programs with many compile defects often have many test defects.





Reviews and Inspections

- The principal focus of inspections should be to find 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
 - produces higher-quality products

Defect Prevention

- Defect prevention is important because
 - it is always expensive to find defects
 - if the defects can be prevented, you can avoid the costs of finding and fixing them
 - defect prevention analysis costs are incurred once, but the savings apply to every project
- Defect prevention should follow an orderly strategy and a defined process.
- For the PSP, defect prevention actions include gathering defect data, improving design methods, and prototyping.

Defect Prevention Strategy -1

- Set priorities for the defect types that are most
 - frequently found
 - troublesome
 - easily prevented
 - annoying
- The defect-prevention process has the following steps.
 - Follow an established schedule.
 - Select one or two defect types for initial action.
 - Measure the effectiveness of defect prevention.
 - Make adjustments and continue.

Defect Prevention Strategy -2

- When setting initial priorities, consider the defect types found most frequently in integration and system test.
- Use PSP data to pick one or two defect types for initial action.
- Don't just try harder; establish explicit prevention actions.
- Incorporate these actions into your process scripts, checklists, and forms.

Messages to Remember

- Improve product quality and accelerate development work by
 - doing reviews and inspections to remove defects before test
 - using testing to check product quality, not to remove volumes of defects
- Design and code reviews
 - 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 practices
