Review Processes

- Quality and Reviews
  - efficacy of Q/A techniques
  - benefits of reviews
  - types of reviews
- Formal Reviews
  - the process
  - the roles
  - risks and how to manage them
- Informal Review Processes
  - differences and trade-offs

Ways to improve quality

- Try to be careful
  - follow established best practices
  - reduce number of mistakes we make
- Peer Reviews
  - get other skilled people to check our work
  - before we do further work based on it
- Testing
  - test for all the problems we can think of
  - try to find the mistakes after we make them
  - go back and fix them before we ship

Reviews

- get other sets of eyes to review our work
  - to find errors and omissions
  - to encourage developers to do better work
- review each major completed work product
  - fix requirements before we do architecture
  - fix architecture before we do the design
  - fix design before we write the code
  - understand how to test code before we write it
  - fix code before we test and ship it
- enabling us to ship better products
  - on-time, with lower development & support costs

Benefits of Reviews

- They can be better than testing
  - finds more problems than testing
  - finds problems sooner and more efficiently
- They are excellent training tools
  - process, methodology, standards, technique
- They improve information dissemination
  - reviewers learn other parts of the product
- They improve programming skills
  - as people learn from others' skills/mistakes

Many types of reviews

- Reviews for almost every project phase
  - Requirements reviews
  - Architectural reviews
  - Design reviews
  - Test Plan reviews
  - Code reviews
- Often used as acceptance criteria
  - before moving on to next project phase
- Different reviews ask different questions
  - but the process remains the same

Requirements Reviews

- Ensuring we are building the right thing
- user-level requirements
  - clear and well justified, widely agreed to
  - traceable and prioritized
  - relatively complete and stable
  - do we believe we can satisfy them?
- validate component-level requirements
  - reasonable, complete, consistent, testable
  - do they add up to the user-level requirements

Architectural Reviews

- Review architecture prior to design
- Is it capable of meeting requirements?
  - embraces all applicable standards
  - no performance or robustness issues
- Will it be practical to build & support?
  - all components well specified, look doable
  - reasonable use of off-the-shelf technology
  - good modularity, well abstracted interfaces
- Is there anything here we'll regret later?

Design Reviews

- Review Design prior to implementation
- Is the design reasonable?
  - it will satisfy all component requirements
  - no major concerns about it working
  - complete, correct, and relatively simple
- Is it clear how to build this component?
  - clearly achievable with existing technology
  - no significant open design questions
- Is the design testable?
  - adequately observable and controllable

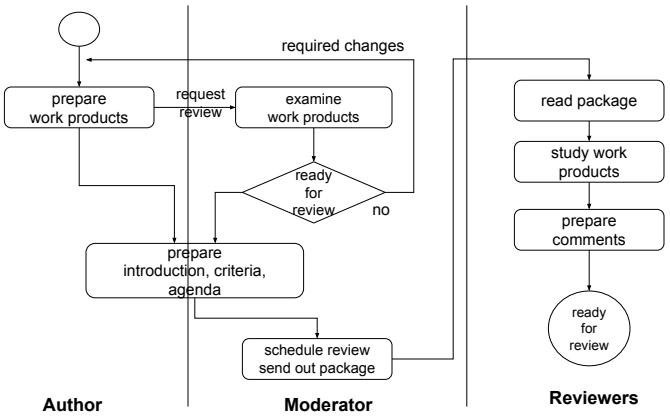
Test Plan Reviews

- Review proposed test cases
  - each clearly and adequately described
  - sufficient to cover all likely problems
  - avoid redundant or useless test cases
- Review proposed testing strategy
  - enables code to be tested as developed
  - clear how all tests will be implemented
  - good use of standard automation technology
- How much confidence will it give us?

Code Reviews

- Review Code (pre-testing vs pre-push)
- Does this code implement the design?
  - implements all specified functionality
  - appropriately handles all reasonable cases
- Is this code obviously correct
  - un-obviousness often hides incorrectness
- Does it conform to applicable standards?
  - naming, commenting, layout conventions
  - portability, tool enabling conventions, etc.

Preparation for a Review



Author

- person who created the work product
- Preparatory tasks
  - prepare the work product for review
    - known problems have already been addressed
    - purpose of review is not to ask for help
  - prepare introductory & background materials
- During a traditional formal review
  - author does not present
  - may answer questions
  - clarify points that have been missed

Review Materials - Introduction

- background
  - what project/component are we discussing
  - what do reviewers need to know about it
    - history, key problems, important decisions, etc.
  - where can they find additional information
    - requirements, designs, issue analyses, tool docn
- goals of this review
  - specific work products will be reviewed
  - scope of this review (what is in/out of bounds)
  - what approval means

Materials - the Work Products

- the work products to be reviewed
  - specifications, designs, code, test plans, etc.
  - these should speak for themselves
  - wasteful to review them before they're ready
- a plan to structure the review
  - a table-of-contents for the work product
  - what will be reviewed, in what order
    - correct order is often critical to understanding
  - what types of issues will be covered when
  - this is the basis for the review agenda

Materials - Approval Criteria

- requirements to be satisfied
  - customer, organizational, standards
- review check-lists
  - many organizations have review check-lists
    - questions to asked, problems to consider
  - they are evolved based on experience
    - e.g. at the end of McConnell's chapters
  - these can help the reviewers
    - by reminding them of things to consider
  - they can't substitute for thought/experience

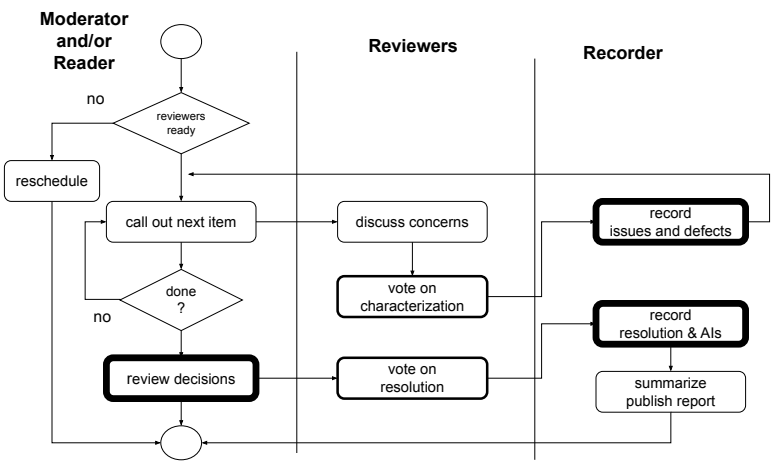
The Review Moderator

- experienced person (other than author)
- Preparatory tasks
  - schedule the review
  - review & distribute the preparatory materials
  - prepare and distribute an agenda
- During the review
  - keep review moving per the agenda (w/o digressions, rat-holes, scope-excursions)
  - ensure all voices heard, no key points lost
  - ensure decisions & action items are agreed to

Reviewer Preparation

- Study all work products
  - study until they are fully understood
    - may be assigned a portion or perspective
    - may need to review background material
  - review against stated scope/goals/criteria
  - consider completeness, possible problems
  - prepare written notes on all concerns
  - this study is where most problems are found
- Why this must be done before review
  - study takes much time and thought
    - could not be done in real-time at the meeting
  - arrive prepared to discuss these issues

The Review Process



Reader

- experienced person (other than the author)
  - could be moderator
- during the review
  - guide the discussion through the code
  - following the prepared review materials
  - calling out each interesting element
  - asking for observations and issues
  - attempting to do these without bias

Reviewers (2-6)

- adequate technical experience
  - all reviewers must understand work products
  - others may attend for training purposes
    - to learn the technology or review process
    - but these people are not there as reviewers
- breadth of relevant expertise
  - familiar w/problem domain, tools, products
- take the process seriously
  - prepare, fully participate, there to help

Recorder

- take notes during the review
  - record all defects discovered
    - it is useful to assign a severity to each
  - record all issues raised
    - questions, suggestions, escalations, etc.
  - record decision and action items
    - accepted, major/minor revisions, further review
  - summarize all decisions at end of review
- publish a report of the review
  - recorder may be a process person, observing process & collecting metrics

Issue/Defect Resolutions

- Issue characterizations:
  - issues (to be resolved)
  - defects (to be fixed)
  - well described (clear, specific, constructive)
- Disposition must be agreed to for each:
  - major (must) … would not work as written
  - minor (should) … measurable improvement
  - advice … things to be considered
  - answered questions have been dealt with

Decision and Report

- At end of meeting
  - review each outstanding defect/issue
    - clearly state the issue
    - confirm the agreed upon status
  - decide/announce the state of the project
    - **approved** (w/enumerated changes)
    - **requires another review**
      - to review non-straight-forward fixes
      - to review proposed resolutions to large issues
      - must give the team a clear path to approval
- These must also be in the report

Potential Problems

- scope issues
  - digressions and rat-holes
  - revisiting past decisions
- productivity issues
  - materials difficult to understand
  - wrong reviewers, insufficient preparation
  - reviewer burn-out
- ego issues
  - discussing people rather than problems
  - telling the author how to write his/her code
  - author needs to defend his/her decisions

Less Formal Processes

- Structured Walk Through
  - conducted by author, perhaps w/o preparation
  - no check lists, moderator, or written report
- Code Reading
  - give code to reviewers (similar to review)
  - reviewers send feedback directly to author
  - no meeting, moderator, or written report
- Pull Request Reviews
  - code owners review, discuss & approve changes
- Pair Programming
  - code reviewed by partner, as it is written

How do they work?

- they can work well with a good author
- advantages
  - fewer people, less overhead, faster
  - simpler products don't require as much review
  - still has potential to find most of the problems
  - can include training and problem solving
- disadvantages
  - fewer people may mean lost perspectives
  - some issues require exploration/discussion
  - less opportunity for emergent insights
  - author may have opportunity to "sell" ideas
  - no scribe may mean lost issues and concerns

Mid-Term Exam (Thu 2/22)

- closed book (no notes), 75 minutes
  - download (from schedule) at 09:30
  - edit and submit as ASCII text
  - up-load (to Canvas) by 11:00
- ten multi-part questions
  - all based on key learning objectives
    - most focusing on concepts and issues
  - all call for brief answers
  - most parts more difficult than quiz questions
    - but have been answered in reading/lectures
- one extra credit question
  - practical application, not discussed in class

For Next Lecture

- Usability and Usability Testing:
  - Wikipedia: Usability Testing
  - usability.net: User Centered Design
  - Talin: User Interface Design Principles
  - Bay: Designing Games that Don't Suck
  - Thompson: Halo 3 – a New Science of Play
- Different Types of User Interfaces:
  - Nielsen: Web vs GUI interfaces
  - Kampe: Content Architecture
  - Kampe: CLI design

Supplementary Slides

Review Scope

- review must be kept at designated level
  - requirements … don't design the system
  - architectural … don't design the components
  - design … don't over-constrain implementor
- avoid descending to lower level issues
  - may be needed to illustrate potential issues
  - suggestions can be made outside the review
- avoid re-specifying/designing the system
  - escalations can be included in the report

Too much ego involvement

- author is too defensive to get input
  - hence author can't moderate/review/record
  - ensure senior engineers set a good example
  - exclude management from review meetings
- reviewers focusing on the wrong things
  - different approaches, style, personality
  - clearly defined review scope
  - written standards and check-lists
  - moderator must manage the egos & scope

Deeper A_2

- How do reviews encourage better work?
  - *If we know that our work will be reviewed by people we respect, we will be strongly motivated to present them with our best work.*
  - *If we observe problems in other peoples' work we will try to avoid them in our own.*
  - *If we observe things that have been done very well in other peoples' work, we are likely to try to emulate their example.*

Deeper A_4

- Why should a work product be "complete" before it is submitted to formal review?

  It is a waste of time to ask other people to find problems that the author already knows about.
- Does this mean you can't get help from your peers until you are mostly done?

  You should feel free to discuss problems with others throughout the process. The purpose of a formal review is to ensure that a work product is complete and correct.

Deeper A_1

- Why do reviewers find problems the author can't find?
  - *The author, in studying the problem, has come to conclusions (and assumptions) that the reviewers have not (yet) made.*
  - *The author may review something and see what (s)he intended, rather than what (s)he actually wrote.*
  - *Different people draw on different perspectives and experiences .*

Deeper A_3

- Why do we call out these particular work products for review?
  - *Each of these work products serves as the basis for significant amounts of future work. This means that errors in these work products will experience defect amplification, and be much more expensive to fix if found later.*
  - *Find mistakes before you base substantial additional work on them.*

Deeper B_1

- Why is testing only 35% to 55% effective at finding defects?
  - *Because testing only covers specifically enumerated situations … and we don't think of everything, and we don't develop test cases for everything we think of.*
  - *Because test cases don't cover missing requirements and specifications.*
  - *Because test plans and test cases are developed by people who are just as fallible as the people who develop the code.*

Deeper B_2

- How do reviews find problems sooner than testing?

  Reviews find many problems before the code is even written.

- Why are reviews more efficient at defect elimination than testing?

  We save the time we would spend writing the code and test cases, running the test cases, debugging the problem, and fixing the code.

Deeper B_3

- If Pair Programming was not an effective training tool, why are reviews better?

  The problem with putting untrained people in programming pairs is that they are unable to substantially contribute to the work (thus defeating the purpose).

  In reviews they are able to learn process, methodology, and technology without slowing down the review process. Even if they only observe, they do not reduce the group's productivity.

  Also, studying code is a great way to learn.

Deeper C_1

- Are there problems with having the author explain the code?

  (a) The code should speak for itself. If it doesn't, it needs to be better commented.

  (b) If the author explains his/her intent for the code, reviewers may buy in to the approach without determining its correctness for themselves.

Deeper D_1

- Why should the scope and goals of the review be explicitly stated?

  (a) So reviewers will know what kinds of issues they are and are not looking for.

  (b) To avoid wasting time preparing or discussing inappropriate comments.

- Why might interesting topics be designated out of scope?

  (a) Because they have already been resolved.

  (b) Because they will be discussed elsewhere.

Deeper D_2

- Why is it important to lay out the order in which pieces will be reviewed?

  (a) So reviewers will know when it is appropriate to raise which issues.

  (b) Because it may be easier to discuss some parts after other parts have already been reviewed.

Deeper E_1

- Why should the moderator be someone other than the author?

  The moderator's primary responsibilities include ensuring that all points of view are heard, and that the focus stays on defects in the code rather than ego issues.

  This requires that moderator to be neutral and dispassionate … which would be difficult for the author.

Deeper E_2

- Why should the reader be someone other than the author?
  The reader is supposed to move the discussion through the code, giving reviewers an opportunity to make comments about each section.
  This author might feel compelled to make his/her own observations about the code during the reading, and to respond to comments raised by other reviewers.

Deeper E_3

- Why is the author not included in the reviewers?
  The author has already had ample opportunity to make observations about, and to make changes to the code. The purpose of the review is to get input from others.

Deeper E_4

- Why should the recorder/scribe be someone other than the author?
  The scribe keeps a record of all issues raised, all action items assigned, and writes the report. This should be done by someone who has no ego in the code or responsibility for its delivery.

Deeper F_1

- How do we recognize and deal with digressions, rat-holes and rehashing old decisions?
  It is the moderator's responsibility to keep the discussion on-target and moving.
  - *If the discussion seems to be digressing, the moderator should bring the discussion back to the components under review.*
  - *When the discussion gets into a rat-hole, the moderator should stop the discussion, note the issue for the report, and move on to the next topic.*
  - *When old issues are being rehashed, the moderator should remind the reviewers what the scope of this review is.*

Deeper F_2

- What should we do if reviewers have difficulty understanding the materials?
  Confusing materials are supposed to be caught by the moderator before the review is scheduled.
  If, however, it becomes clear that the materials are too confusing (or that the wrong reviewers have been selected) the moderator should immediately end the review. The materials should be revised and/or new reviewers should be selected, and the review should be rescheduled.

Deeper F_3

- What should we do if it becomes clear that one or more reviewers have not done the necessary study and preparation?
  The moderator should ascertain the preparedness of the reviewers at the start of the review.
  Unprepared reviewers should be immediately dismissed from the review. If there are not enough reviewers to complete the review, the session should be cancelled and rescheduled.

Deeper

- What should we do if the reviewers get burned out from too long a review sessions?

Review sessions should be limited to 1-2 hours maximum. If a component is too complex to be reviewed in that time, it should be broken up into smaller sub-units for review purposes.

Deeper

- What should we do if the reviewers get into arguments about issues and approaches to a problem?

The moderator should remind the reviewers that the purpose of the review is not to solve problems, but to identify them. Fixing problems is the responsibility of the author, and discussions about how to fix them should be conducted outside of the review meeting.

Deeper

- Why do we limit discussion to defects, and not allow discussion of solutions?

(a) *Because defects are relatively objective, whereas approaches may be more matters of opinion.*
(b) *Because responsibility for solving the problem is best left with the person who is responsible for solving the problem. Second guessing them is highly counter-productive.*
(c) *Because discussions of approaches to solutions can easily turn into long arguments.*
(d) *Because a code review is not an efficient format for brain-storming about how to solve problems.*

Deeper

- The author gets defensive about his/her code?

The moderator should keep the review focused on objective issues and criteria, rather than on opinions about how things should be done.
The presence of management in reviews can increase defensiveness … hence the recommendation that they be excluded.
If the author cannot help but be defensive, they do not have to be present for the review … since they are not an active participant.