SWINBURNE UNIVERSITY OF TECHNOLOGY

SWE20001
Managing Software Projects

Lecture 4a

Quality Review

Principal References

- Ian Sommerville, *Software Engineering* (8th Edition), Addison-Wesley, 2004, Chapter 27.
- Roger S. Pressman, *Software Engineering - A Practitioners Approach* (6th Edition), McGraw-Hill, 2005, Chapter 26.
- Bob Hughes and Mike Cotterell, *Software Project Management* (4th Edition), Wiley, 2006, Chapter 12.
- Carlo Ghezzi, Mehdi Jazayeri, Dino Mandroli, *Fundamentals of Software Engineering* (2nd Edition), Prentice-Hall 2003, Chapter 2.

Quality Review

- Performed by a group of people who carefully ***examine*** part or all of a software system and its associated documentation
- To assess the quality of some artefact against pre-defined quality standards
  - Software or documents may be “signed off” at a review
  - Progress to the next development stage is thereby approved
- Review results should be recorded and maintained
- Automated assessment may contribute to a review

Types of Quality Reviews …

Review type	*Principal purpose*
Formal Technical Reviews (aka design or program inspections)	Driven by a *checklist* ■ detect detailed errors in any product ■ mismatches between requirements and product ■ check whether standards have been followed
Progress reviews	Driven by *budgets, plans and schedules* ■ check whether project runs according to plan ■ requires precise milestones ■ both a process and a product review

Roles in Review Meetings

- **Moderator**: manages the process and facilitates the inspection
- **Scribe**: records the findings during inspection meeting
- **Inspector / Reviewer**: identifies faults in the materials
- **Author**: produces the materials

☞ An Author can never be Moderator, Scribe, or Inspector!

☞ Author is present chiefly to answer questions

Review Meetings

Review meetings should:

- typically involve *4-5 people* (moderator, author, scribe, reviewers)
- require a maximum of *2 hours advance preparation*
- *last at most 2 hours*
- require a maximum of *1 hour* to write a review report

The review meeting should conclude whether the artefact under review is:

- *Accepted* without modification 

- *Provisionally accepted*, subject to corrections (no follow-up review)
- *Rejected*, subject to corrections and follow-up review 

Review Minutes

The review report should *summarize*:

1. *What* was reviewed
2. *Who* reviewed it?
3. What were the findings and conclusions?

The review should also contain a list of identified defects:

1. Number (ie label) of defect
2. What is the defect and where is it in artefact
3. What kind of defect is it? (according to a pre-defined classification scheme)
4. Criticality of defect (e.g., low, medium, high, critical)

☞ *Conclusions should reflect list of identified defects!*

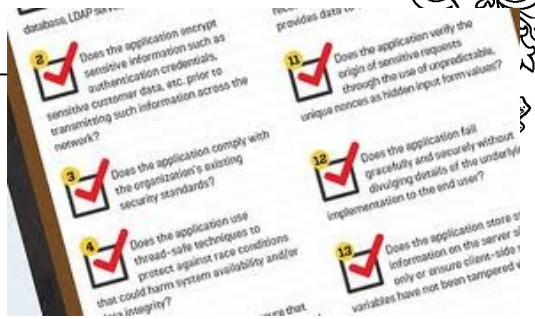
Review Guidelines

1. Review the *product*, not the producer
2. Set an *agenda* and maintain it
3. *Limit debate* and rebuttal
4. *Identify problem areas*, but don't attempt to solve every problem noted
5. Take *written notes*
6. *Limit the number of participants* and insist upon advance preparation
7. Develop a *checklist* for each product that is likely to be reviewed
8. *Allocate resources* and time schedule for reviews
9. Conduct meaningful *training* for all reviewers
10. *Review* your early reviews

Reviews vs. Audits

- A *quality review* verifies whether a given artefact contains the required information, meeting the pre-defined quality requirements, to continue with the development process.
- An *audit* checks whether the quality assurance standards defined by a team (e.g., in a SQAP) are adhered to in their development process.
- (For example, an audit might check that required quality reviews have actually taken place, and been documented correctly!)

Checklists



- Reviews should generally be checklist driven!
  - Well, many people think so, but there are other points of view (eg http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER07/Nwer07-fogelstrom.pdf)
- The wisdom of extensive past experience is brought to bear on the current situation
- Nevertheless, there must remain the option to “discover” something unexpected, and to note this!
- http://www.goldensegroupinc.com/product-design-quality-software-review-procedures-checklists.shtml is a useful reference regarding checklists in the wider sphere

Sample Review Checklists (I)

Software Project Planning

1. Is software scope unambiguously defined and bounded?
2. Are resources adequate for scope?
3. Have risks in all important categories been defined?
4. Are tasks properly defined and sequenced?
5. Is the basis for cost estimation reasonable?
6. Have historical productivity and quality data been used?
7. Is the schedule consistent?

...

This is not comprehensive!

Sample Review Checklists (II)

Requirements Analysis

1. Is information domain analysis complete, consistent and accurate?
2. Does the data model properly reflect data objects, attributes and relationships?
3. Are all requirements traceable to system level?
4. Has prototyping been conducted for the user/customer?
5. Are requirements consistent with schedule, resources and budget?

...

And neither is this, or the others to follow!

Sample Review Checklists (III)

Design

1. Has modularity been achieved?
2. Are interfaces defined for modules and external system elements?
3. Are the data structures consistent with the information domain?
4. Are the data structures consistent with the requirements?
5. Has maintainability been considered?

...

Sample Review Checklists (IV)

Code

1. Does the code reflect the design documentation?
2. Has proper use of language conventions been made?
3. Have coding standards been observed?
4. Are there incorrect or ambiguous comments?

...

Sample Review Checklists (V)

Testing

1. Have test resources and tools been identified and acquired?
2. Have both white and black box tests been specified?
3. Have all the independent logic paths been tested?
4. Have test cases been identified and listed with expected results?
5. Are timing and performance to be tested?

…

Review Results

Comments made during the review should be *classified*

- ***No action***
  - ☐ No change to the software or documentation is required
- ***Refer for repair***
  - ☐ Designer or programmer should correct an identified fault
- ***Reconsider overall design***
  - ☐ The problem identified in the review impacts other parts of the design

Requirements and specification errors may have to be referred to the client.

Quality Reviews in Scrum

High Level Review (Product based)

- Sprint Review (aka Product Review)
- Sprint Retrospective (aka Process Review)

Low Level (Code based)

- Errors in code
  - Automated testing (proactive)
  - Write test before code (proactive)
  - Test driven development (proactive)
  - Pair programming (proactive)
  - Refactoring (proactive)
  - Debugging (reactive)
- Readability of code
  - Pair programming
  - Refactoring
- Integration testing (“DoD” ?)
- Performance testing (“DoD” ?)
- …

DoD = Definition of Done

Sprint Review (aka Product Review)

- Review the “completed” (based on team’s belief) items to the stakeholders
- “Basically a UAT”
- Note the Definition of Done

Sprint Retrospective (aka Process Review)

- Review the items that cannot be completed in the sprint
  - Identify what is wrong in the sprint that causes the team not being able to “complete” these items; and
  - Suggest things the team can do to improve the situations in the next sprint
- Review the process in the sprint
  - Identify the issues that make the current process not working; and
  - Suggest ways the team can do to address those issues in the next sprint

Quality Management in XP

Product

Planning Game

Process

Simple Design

System Metaphor

Teamwork

Continuous Integration

Coding

Testing

Refactoring

Pair programming

Collective Code Ownership

Coding Standards

40 Hour Week

On-Site Customer

Small Releases

What you should know!

- How should you organize and run a review meeting?
- What information should be recorded in the review minutes?