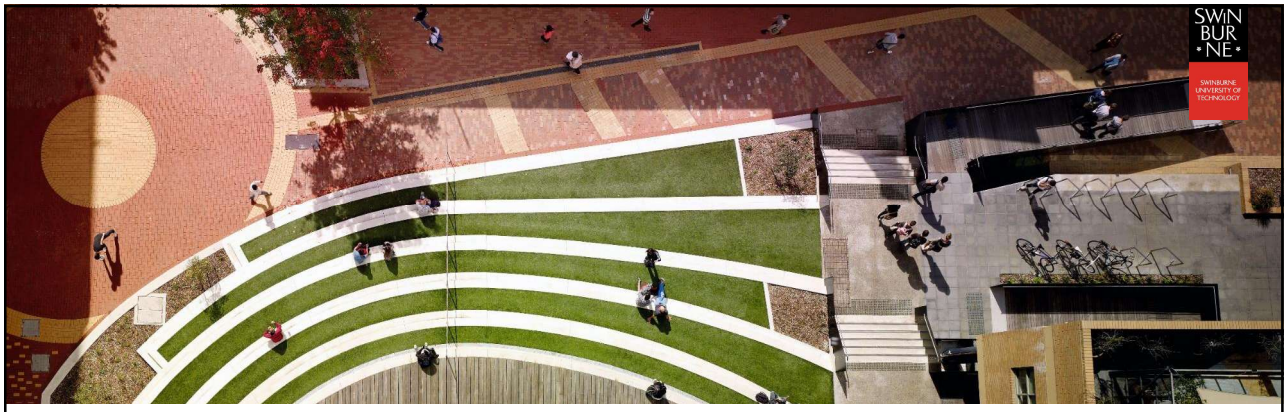


COS80022 – Software Quality and Testing

Week 10 – Review, Inspection and Walkthrough

Outline

- Quality assurance
- Static testing
- Reviews

Software Quality

Software Quality

QA vs. QC

- Quality assurance: process oriented
  - Make sure you are doing the right thing, the right way
  - Defect prevention
  - Process checklists, project audits and methodology and standards development
- Quality control: product oriented
  - Make sure the results of what you've done are what you expected
  - Defect identification
  - Inspection, deliverable peer reviews and the testing process

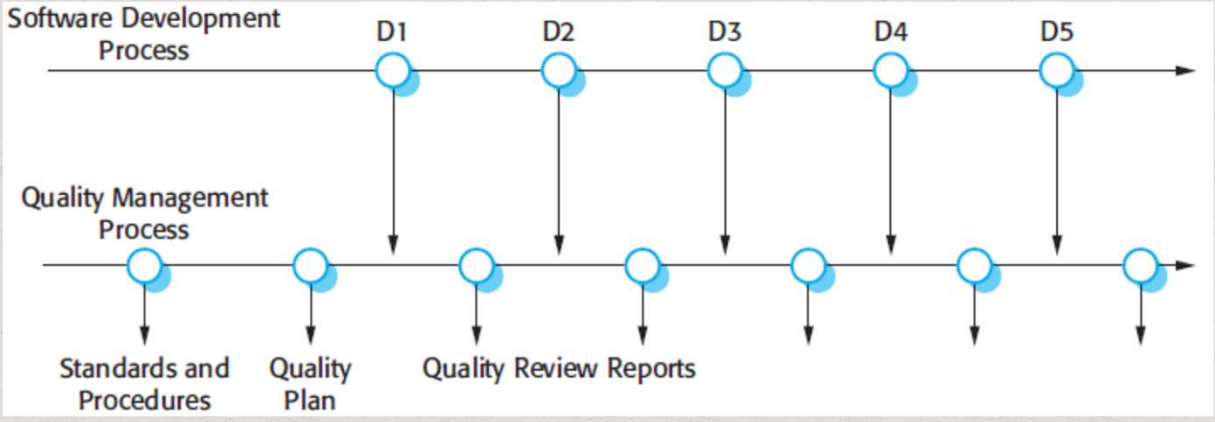
5

QA

- Processes and standards leading to high-quality products
- Introduction of quality processes into manufacturing
- Verification and validation
- Processes of checking that quality procedures have been properly applied

6

QA



QA

- QA team should be independent from development team
  - Objective view
  - Not influenced by development issues
- QA team should have organization-wide responsibility

QA

- Quality plan
  - Product introduction
  - Product plans
  - Process description
  - Quality goals
  - Risks and risk management

QA

- Can software quality be achieved through prescriptive processes based around organization standards and associated quality procedures?
  - Embody good software engineering practice
  - Quality culture

QA

- Different from QA in traditional manufacturing
  - Lack of complete and unambiguous specifications
  - Compromise among different stakeholders
  - Immeasurable quality characteristics
- Subjective process
  - How to judge “acceptable”

QA

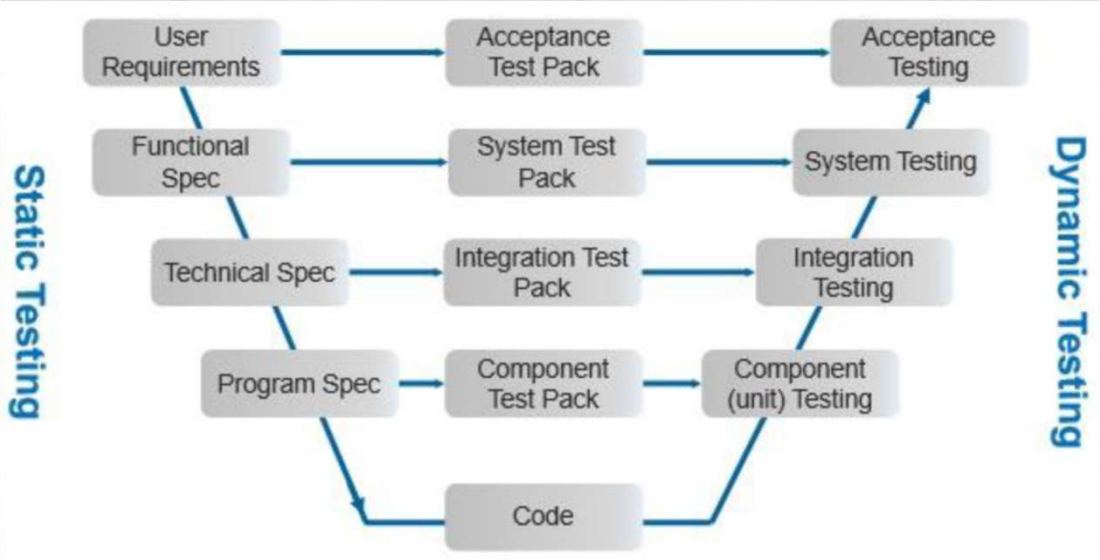
- Have programming and documentation standards been followed in the development process?
- Has the software been properly tested?
- Is the software sufficiently dependable to be put into use?
- Is the performance of the software acceptable for normal use?
- Is the software usable?
- Is the software well structured and understandable?

QA

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

- Impossible to achieve all attributes
- Define the most important attributes
- Trade-off, compromise

Static Testing



Static Testing – Concept & Types

- Testing of a component or system at specification or implementation level without execution of that software
  - Reviews: Manual examination
  - Static analysis: Automated analysis

Static Testing – Work Products

- Specifications
- User stories
- Architecture
- Code
- Testware
- User guides
- Web pages
- Project plans
- Models

Static testing - Benefits

- Early defect detection & correction
- Identify defects not easily found by dynamic testing
- Prevent design or code defects by identifying requirements issues
- Increase development productivity
- Reduced development time and costs
- Testing times and cost reductions
- Reduce total cost of quality cover software's lifetime
- Improved communication within team

Static Testing – Typical Defects

- Requirements defects
- Design defects
- Coding defects
- Deviations from standards
- Incorrect interface specifications
- Security vulnerabilities
- Problems with test coverage
- Insufficient maintainability

Reviews and Inspections

- QA activities checking quality of product delivery
  - Examining the software, documentation, records
  - Used alongside testing as part of V&V
- Purpose: improve software quality
  - Not to assess the performance of development team

Reviews

- A group of people examine the software and associated documentation
  - Potential problems
  - Non-conformance with standards
- Informed judgments about quality level
  - Planning decision, resource allocation

Review

- A type of static testing:
  - An evaluation of software elements or project status to ascertain discrepancies from planned results and to recommend improvement (IEEE 1028 Standard for Software Reviews)
- Objective: Not just to find defects, but also, to find defects earlier in the life cycle and to remove the causes from the process.

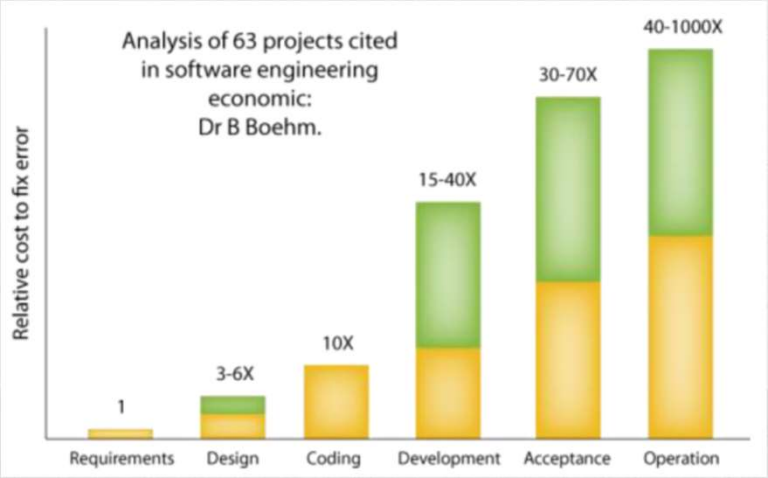
Review

- Early defects are often the most important
  - Defects have the characteristic to multiply themselves top-down;
  - Cost of rework rises exponentially.

60% of the defects have already been made before coding/implementation has started.

Review

- Cost of fixing errors escalates as we move project towards field use



Reviews

- Based on documents
  - Specifications
  - Designs
  - Code
  - Process models
  - Test plans
  - Configuration management
  - Process standards
  - User manuals

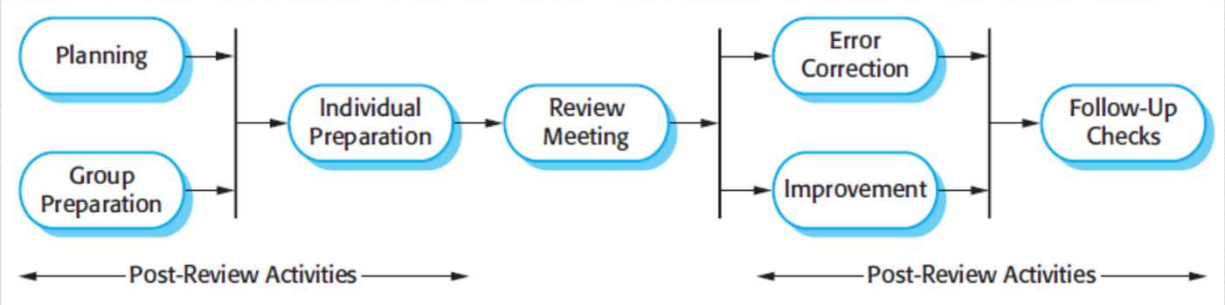
Reviews

- Consistency and completeness
- Conformance to standards
- Problems and omissions

Review Process

- Pre-review activities
  - Planning
  - Group preparation
  - Individual preparation
- Review meeting
- Post-review activities
  - Error correction
  - Improvement
  - Follow-up checks

Review Process



Review Process

- Review team
  - Core of three to four principal reviewers
  - Senior designer: significant technical decisions
  - Invite other project members
  - Circulate the document
  - Ask for written comments
  - Project manager

Review Process

- Development team
  - Colocated and available
  - Document editing tools
  - Annotation
  - Visible comments

Review Process

- In agile: informal review
- Scrum: review meeting every iteration
- In XP: code reviewed constantly
- Not standards-driven
- Detailed quality management procedures slow down the pace of development

Review Process

- Reviews vary from informal to formal
- The formality of a review process is related to a variety of factors
- The focus of a review depends on its objectives

Work Product Review Process

- Planning
- Initiate Review
- Individual Review
- Issue Communication and Analysis
- Fixing and Reporting

Planning

- Defining the scope (e.g. purpose, what document, etc.)
- Estimating effort and timeframe
- Identifying review characteristics such as the review type
- Selecting the people to participate in the review and allocating roles
- Defining the entry and exit criteria for more formal review types (e.g. inspections)
- Checking that entry criteria are met (for more formal review types)

33

Initiate Review

- Distributing the work product and other material (e.g. checklists)
- Explaining the scope, objectives, process, roles, and work products to the participants
- Answering any questions that participants may have about the review

34

Individual Review

- Also known as individual preparation
- Reviewing all or part of the work product
- Noting potential defects, recommendations, and questions

Issue Communication & Analysis

- Communicating identified potential defects (e.g. in a review meeting)
- Analysing potential defects, assigning ownership and status to them
- Evaluating and documenting quality characteristics
- Evaluating the review findings against the exit criteria to make a review decision

Fixing & Reporting

- Creating defect reports from those findings that require changes
- Fixing defects found in the work product reviewed
- Communicating defects to the appropriate person or team
- Recording updated status of defects (in formal reviews), potentially including the agreement of the comment originator
- Gathering metrics (for more formal review types)
- Checking that exit criteria are met (for more formal review types)
- Accepting the work product when the exit criteria are reached

Roles

Purpose of Reviews

- One the main objectives is to uncover defects
- All review types can aid in defect detection
- Selected review type should be based on the needs of the:
  - Project
  - Available resources
  - Product types and risks
  - Business domain
  - Company culture

Types of Review

- Informal review
- Walkthrough
- Technical review
- Inspection

Informal Review

- Main characteristics:
  - Not based on a formal process
  - May not involve a review meeting
  - May be performed by a colleague of the author or by more people
  - Results may be documented
  - Varies in usefulness depending on the reviewers
  - Use of checklists is optional
  - Very commonly used in agile development

Informal Review

- May also be referred to as buddy check, pairing, or pair review
- Main purpose:
  - detecting potential defects

Technical Review

- Main characteristics:
  - Reviewers should be technical peers of the authors, and technical experts in the same or other disciplines
  - Individual preparation before the review meeting is required
  - Review meeting is optional, ideally led by a trained facilitator
  - Scribe is mandatory, ideally not the author
  - Use of checklists is optional
  - Potential defect logs and review reports are typically produced

Technical Review

- Main purposes:
  - Gaining consensus
  - Detecting potential defects

Walkthrough

- Main characteristics:
  - Individual preparation before the review meeting is optional
  - Review meeting is typically led by the author of the work product
  - Scribe is mandatory
  - Use of checklists is optional
  - May take the form of scenarios, dry runs, or simulations
  - Potential defect logs and review reports may be produced
  - May vary in practices from quite informal to very formal

Walkthrough

- Main purposes:
  - Detecting potential defects
  - Improving the software product
  - Considering alternative implementations
  - Evaluating conformance to standards and specifications

Inspection

- Main characteristics:
  - Follows a defined process with formal documented outputs
  - Uses clearly defined roles and may also include a dedicated reader
  - Individual preparation before the review meeting is required
  - Reviewers are peers of the author or experts in other relevant disciplines
  - Specified entry and exit criteria are used
  - Scribe is mandatory
  - Review meeting is led a trained facilitator (not the author)
  - Author cannot act as the review leader, reader, or scribe
  - Potential defect logs and review report are produced
  - Metrics are collected and used to improve the entire software development process, including the inspection process

Inspection

- Main purpose:
  - Detecting potential defects
  - Evaluating quality and building confidence in the work product
  - Preventing future similar defects through author learning and root cause analysis

Program Inspections

- "peer review"
  - Team members collaborate to find bugs
  - Complement testing: no program execution
- Team members from different backgrounds
  - Careful line-by-line review
- Look of defects and problems
  - Logical errors
  - Anomalies
- Checklist

Program Inspections

- Data faults
  - Are all program variables initialized before their values are used?
  - Have all constants been named?
  - Should the upper bound of arrays be equal to the size of the array or (Size-1)?
  - If character strings are used, is a delimiter explicitly assigned?
  - Is there any possibility of buffer overflow?

Program Inspections

- Control faults
  - For each conditional statement, is the condition correct?
  - Is each loop certain to terminate?
  - Are compound statements correctly bracketed?
  - In case statements, are all possible cases accounted for?
  - If a break is required after each case in case statements, has it been included?

Program Inspections

- Input/output faults
  - Are all input variables used?
  - Are all output variables assigned a value before they are output?
  - Can unexpected inputs cause corruption?

Program Inspections

- Interface faults
  - Do all function and method calls have the correct number of parameters?
  - Do formal and actual parameter types match?
  - Are the parameters in the right order?
  - If components access shared memory, do they have the same model of the shared memory structure?

Program Inspections

- Storage management faults
  - If a linked structure is modified, have all links been correctly reassigned?
  - If dynamic storage is used, has space been allocated correctly?
  - Is space explicitly deallocated after it is no longer required?
- Exception management faults
  - Have all possible error conditions been taken into account?

Program Inspections

- In agile: no formal inspection
- Rely on team members
  - Cooperating to check each other's code
- Informal guidelines
- XP: pair programming

Use of Review Types

- A software product or related work product may be the subject of more than one review
- If more than one type of review is used, the order may vary
- The types of defects found in a review vary, depending especially on the work product being reviewed

Review Techniques

- Ad hoc
- Checklist-based
- Scenarios and dry-runs
- Role-based
- Perspective-based

Ad Hoc Technique for Review

- Reviewers provided with little guidance on how to review
- Typically read work product sequentially, raising issues as found
- Commonly used technique needing little preparation
- Highly dependent on skills of reviewer

Checklist-Based Technique for Review

- Detect issues based on checklist
- Checklists distributed at review initiation
- Set of questions based on potential defects
- Derived from experience
- Specific to type of work product
- Maintain regularly to keep relevant
- Systematic coverage of typical defect types
- Also look for defects outside the checklist

59

Using Checklists

- A list of questions
- A negative answer identifies an error
- Interpret the rules
- Identify frequently made mistakes
- Must be maintained

60

Using Checklists

- Example: We review a work product for testability.
- Can we design tests from it? Is it:
  - Clear?
  - Complete?
  - Concise?
  - Unambiguous?
  - Measurable?
- "The software should be very user friendly". **X**
- "The software must conform to the usability standards stated in the usability standards document". √

Using Checklists

- Does this document conform to your standards and house-style?
- Is this document clear and unambiguous to the intended readership?
- Are all statements consistent with other statements in the same or related documents?

Using Checklists

- Is this document complete? (or has something vital been missed out);
- Have irrelevant topics been left out?
- Are the diagrams and pictures clear and a useful aid in explaining the text?
- Have you checked the references outside the document?
- Test Analyst is more likely to be reviewing requirements, use cases, user stories or user interfaces, rather than code or architecture;
- Checklists will vary for each of these product types.

Requirements Checklist

- Testability of each requirement
- Acceptance criteria for each requirement
- Availability of a use case calling structure, if applicable
- Unique identification of each requirement/use case/user story
- Versioning of each requirement/use case/user story
- Traceability for each requirement from business/marketing requirements
- Traceability between requirements and use cases

Use Case Checklist

- Is each field and its function defined?
- Is the main path (scenario) clearly defined?
- Are all alternative paths (scenarios) identified, complete with error handling?
- Are the user interface messages defined?
- Is there only one main path (there should be, otherwise there are multiple use cases)?
- Is each path testable?

User Story Checklist

- Is the story appropriate for the target iteration/sprint?
- Are the acceptance criteria defined and testable?
- Is the functionality clearly defined?
- Are there any dependencies between this story and others?
- Is the story prioritized?
- Does the story contain one item of functionality?

User Interface Checklist

- Is each field and its function defined?
- Are all error messages defined?
- Are all user prompts defined and consistent?
- Is the tab order of the fields defined?
- Are there keyboard alternatives to mouse actions?
- Are there "shortcut" key combinations defined for the user (e.g., cut and paste)?
- Are there dependencies between fields (such as a certain date has to be later than another date)?
- Is there a screen layout?
- Does the screen layout match the specified requirements?
- Is there an indicator for the user that appears when the system is processing?
- Does the screen meet the minimum mouse click requirement (if defined)?

Adapting Checklists

- Standard checklists can be tailored:
  - Organisation (e.g. company policies standards, conventions);
  - Project (e.g. risks, technical standards);
  - Object under review (e.g. Programming language).
- Checklists promote discussions.
- As a result they may be enhanced.

Scenarios and Dry-Runs for Review

- Provided with structured guidelines on how to read through the work product
- Supports dry runs based on expected usage
- Provides guidelines on how to identify specific defect types
- Reviewers should not be constrained by documented scenarios

Scenarios

- Scenario is a narrative description of what people do and experience as they try to make use of computer systems and applications
  - Concrete, focused, informal description of a single feature of the system from the viewpoint of a single actor
  - Cannot replace use case
- Use cases can be identified based on scenarios
- Reversely, scenarios can be used when reviewing use cases

Scenarios for review

Use case name	WithdrawCash
Participating actors	Initiated by *BankCustomer* Operated on *ATMSystem* Communicates with *BankNetwork*
Flow of events	1. *BankCustomer* activates the *WithdrawCash* function of the *ATMSystem*. 2. *ATMSystem* displays a prompt to select an account to withdraw from (cheque, savings, or credit). 3. *BankCustomer* selects the account they would like to withdraw cash from. 4. *ATMSystem* prompts for an amount to be withdrawn. 3. *BankCustomer* enters the amount they would like to withdraw in the provided field. 4. *ATMSystem* communicates with the *BankNetwork* to validate that the withdraw amount is within the current balance for the selected account. *ATMSystem* then begins dispensing the money. A prompt message asking to print a receipt is displayed. 5. *BankCustomer* collects the cash dispensed by the *ATMSystem* and selects whether to print a receipt. 6. *ATMSystem* communicates with the *BankNetwork* to update the balance on the account. *ATMSystem* displays that the transaction is complete and prints a receipt.

Scenarios for review

- Scenario 1: A customer successfully withdraws cash
  - Review: Scenario covered (Steps 1-6)
- Scenario 2: A customer fails to withdraw cash due to insufficient deposit
  - Review: Scenario not covered (no alternative path when the account does not have enough amount)

Role-Based Technique for Review

- Evaluate the work product from the perspective of different roles
- Roles could be end user types, e.g. experienced or inexperienced
- Roles could be organisation based, e.g. system admin or performance tester

Perspective-Based Technique for Review

- Review done from perspective of different stakeholders
- Different viewpoints give more in-depth review and test duplication of issues raised
- Reviewers use the work product under review to generate items that would be derived from it
- Checklists are used to guide reviewer
- Most effective technique for reviewing requirements and technical work products

Success Factors for Review

- To have a successful review, the appropriate type of review and the techniques used must be considered
- There are a number of organisational and people-related factors that will affect the outcome of the review

Organisational Success Factors

- Each review has clear objectives
- Review types applied are suitable to achieve the objectives and are appropriate to the type of work product
- Any review techniques used, are suitable for effective defect identification
- Any checklists used address the main risks and are up to date
- Large documents are written and reviewed in small chunks, so that quality control is exercised by providing authors early and frequent feedback on defects
- Participants have adequate time to prepare
- Reviews are scheduled with adequate notice
- Management supports the review process

People-Related Success Factors

- The right people are involved to meet the review objectives
- Testers are seen as valued reviewers who contribute to the review
- Participants dedicate adequate time and attention to detail
- Reviews are conducted on small chunks, to ensure concentration
- Defects found are acknowledge, appreciated, and handled objectively
- The meeting is well-managed
- The review is conducted in an atmosphere of trust
- Participants avoid body language and behaviours that might indicate boredom, exasperation, or hostility to other participants
- Adequate training is provided, especially for more formal review types such as inspections
- A culture of learning and process improvement is promoted

Summary

- Static testing
  - Work products that can be examined by static testing
  - Benefits of static testing
  - Differences between static and dynamic testing
- Review
  - Success factors for reviews
  - Applying review techniques
  - Review types
  - Roles and responsibilities in a formal review
  - Work product review process

References

- Sommerville, I. *Software Engineering* (Chapter 24)
- International Software Testing Qualifications Board. ISTQB Foundation Level (Core) Syllabus

Next

- Tutorial
  - Static testing
- Next week
  - Non-functional testing