
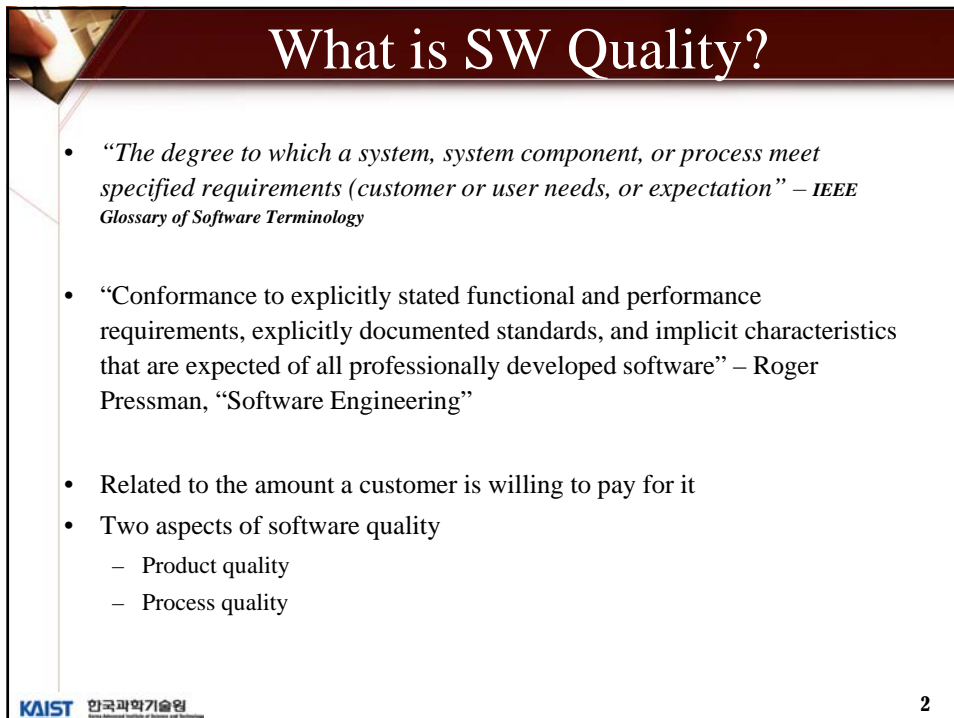


Software Engineering Economics (CS656)

Software Quality Management

Jongmoon Baik

What is SW Quality?

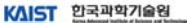
- “The degree to which a system, system component, or process meet specified requirements (customer or user needs, or expectation)” – IEEE Glossary of Software Terminology
- “Conformance to explicitly stated functional and performance requirements, explicitly documented standards, and implicit characteristics that are expected of all professionally developed software” – Roger Pressman, “Software Engineering”
- Related to the amount a customer is willing to pay for it
- Two aspects of software quality
 - Product quality
 - Process quality

KAIST 한국과학기술원
Korea Advanced Institute of Science and Technology

2

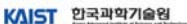
Quality Mgmt. Emcompasses:

- Software Quality Assurance (SQA)
- Specific quality assurance and quality control tasks
 - Formal technical reviews and a multitiered testing strategy
- Effective software engineering practices
 - Methods and tools
- Control of all software work products and the changes made to them
- Procedure to ensure compliance with software development standards (when applicable)
- Measurement and reporting mechanisms

 한국과학기술원 3

Quality of Design vs. Quality of Conformance

- Quality of Design – “the characteristics that the designers specify for an item”
 - Requirements, specifications, the design of the system
- Quality of Conformance – “the degree to which the design specifications are followed during manufacturing”
 - An issue focused primarily on implementation

 한국과학기술원 4

How to Achieve Software Quality?

The diagram illustrates the components of software quality assurance. At the top, a box labeled 'Software Life Cycle' contains a yellow box labeled 'SQA'. Inside the 'SQA' box, there is a large orange oval labeled 'V&V' and a smaller green oval labeled 'S/W Testing'. Below this, a silhouette of two hands holding up a box labeled 'Software Engineering Methods' is shown. This box contains a list of methods: 'Introduce more precision into the dev. process', 'Modern Dev. Techniques', 'Software Peer Reviews', 'Test effort reduction by modular design', and 'etc.'.

Software Life Cycle

SQA

V&V S/W Testing

Software Engineering Methods

- Introduce more precision into the dev. process
- Modern Dev. Techniques
- Software Peer Reviews
- Test effort reduction by modular design
- etc.

KAIST 한국과학기술원
5

Software Quality Assurance

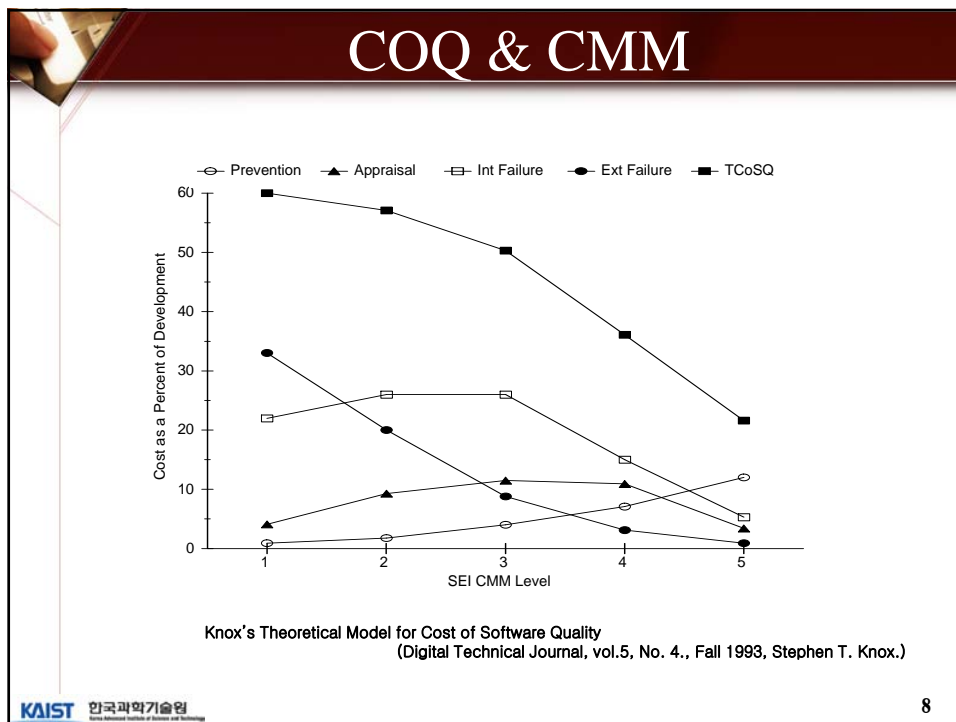
- A set of auditing and reporting functions
 - To assess the effectiveness and completeness of quality control activities
- Primary goal of SQA
 - To provide management with data necessary to be informed about product quality
 - Gain insight and confidence about whether product quality is meeting its goal

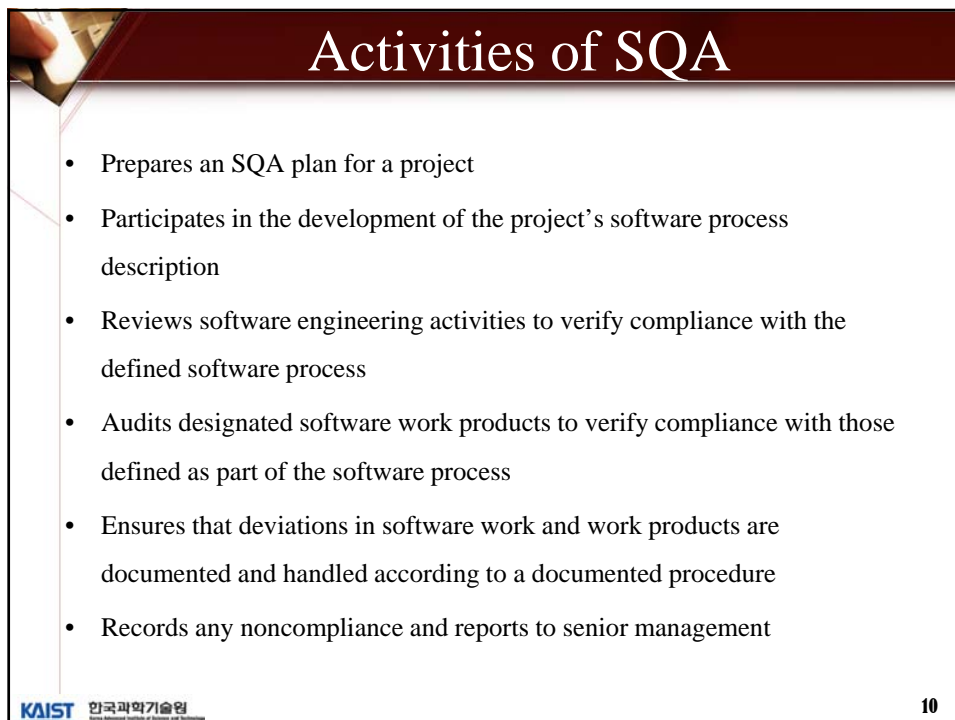
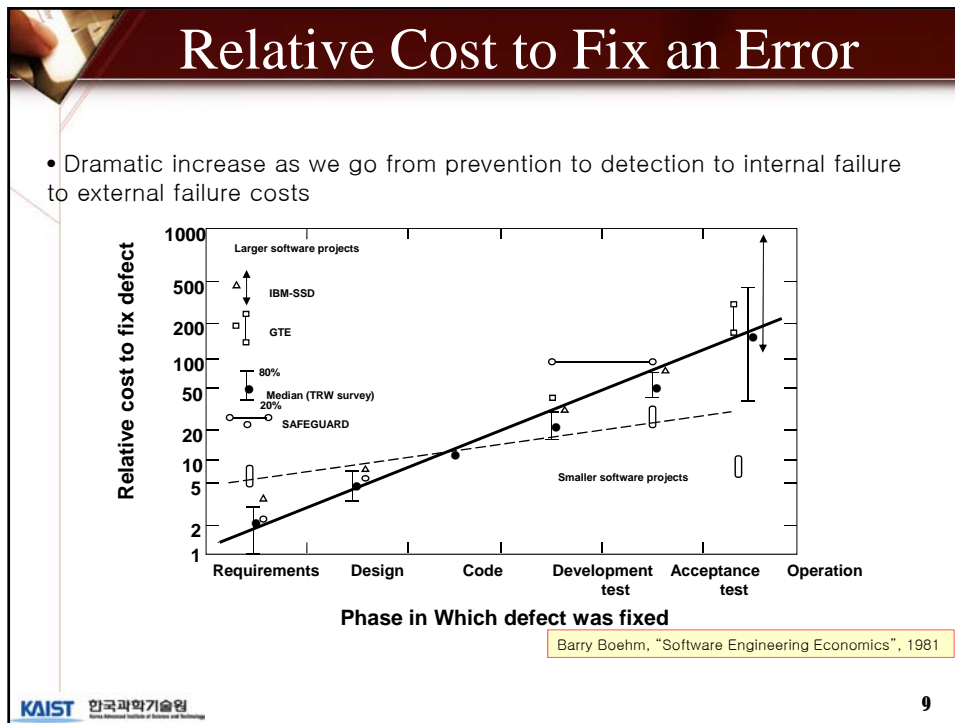
KAIST 한국과학기술원
6

Cost of Quality

- “All costs incurred in the pursuit of quality or in performing quality-related activities”
 - Provide a baseline for the current cost of quality
 - Identify opportunities for the quality cost reduction
 - Provide a normalized basis of comparison (on a dollar basis)
- Four types of Quality Cost
 - Prevention Cost
 - Quality planning, formal technical reviews, test equipments, and training
 - Appraisal Cost
 - In-process and inter-process inspection, equipment calibration, maintenance, and testing
 - Internal Failure Cost
 - Rework, repair, and failure mode analysis
 - External Failure Cost
 - Complaint resolution, product return and replacement , help line support, and warranty work

7





What is a Peer Review?

- “A type of software review in which a work product is examined **by its author and/or one or more colleagues of its author**, in order **to evaluate its technical content and quality**.” - Wikipedia
 - Software Project Plan, Requirement Specification, Design, Code, Test Process, Test Case, etc.
- “The review of work products performed **by peers** during development of the work products **to identify defects for removal**.”
 - CMMI Guidelines for Process Integration and Product Improvement, Addison Wesley, 2003

KAIST 한국과학기술원
11

Objectives of Review

- To verify the work product meets requirements
- To detect and remove defects from the software work products early and efficiently
- To gain confidence in work products
- To prevent their leakage into field operations
- To reduce risks

KAIST 한국과학기술원
12

Benefits of Software Peer Reviews

- **Get another perspective**
 - Finding defects can be easier for someone who hasn't seen the artifact before and doesn't have preconceived ideas about its correctness
- **Knowledge transfer about:**
 - software artifacts and defect detection
- **Detect and Remove errors early**
 - Dramatic cost reduction to fix them
- **Reduce rework and testing effort**
 - overall development effort reduction

KAIST 한국과학기술원
KAIST Korea Advanced Institute of Science and Technology

13

Formal/Informal Review

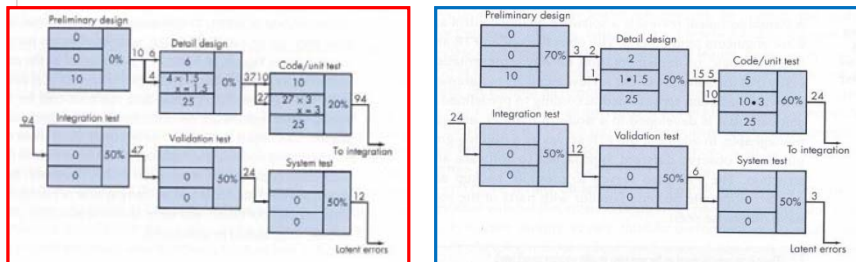
- **Formal Review:**
 - Highly structured process for preparation, meeting protocol, data collection, and post meeting activities
 - Table-top meeting oriented
 - Excellent for ensuring final product quality and standards
 - Performed on complete artifacts
 - e.g.: Software Fagan Inspection, Formal Technical Review...
- **Informal Review:**
 - Less structure, formality, and rigidity
 - More open discussion (peer review)
 - Table-top or presentation oriented
 - Excellent for initial artifact presentation, artifact discussion, or selection of alternatives
 - Performed on incomplete artifacts
 - e.g.: Walkthrough...
- **Both formal and informal inspections have their place in a project**

KAIST 한국과학기술원
KAIST Korea Advanced Institute of Science and Technology

14

Defect Amplification & Removal

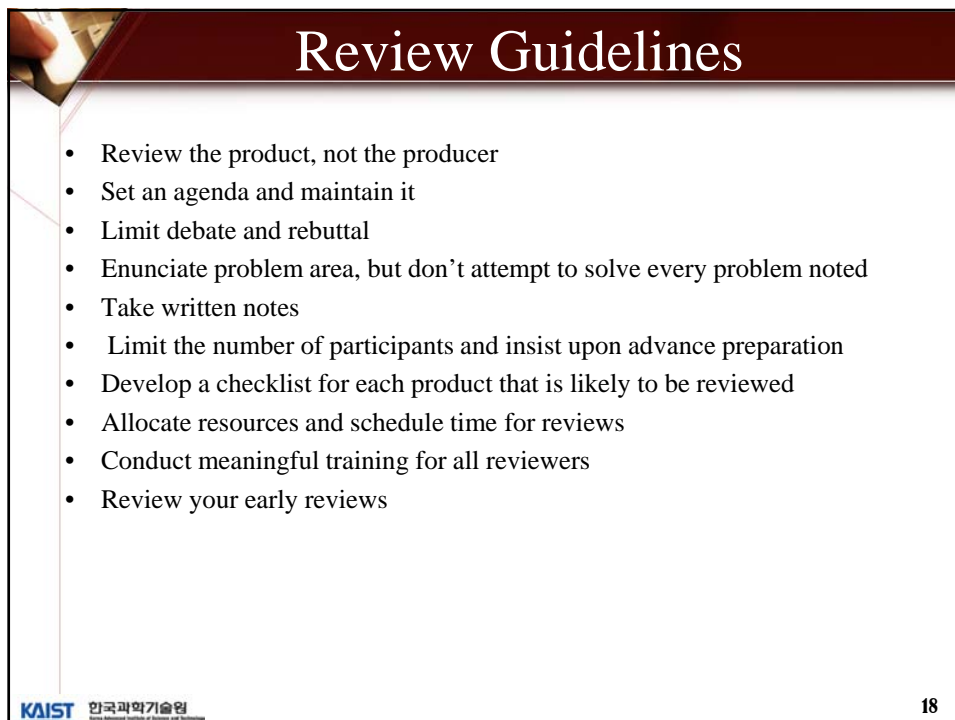
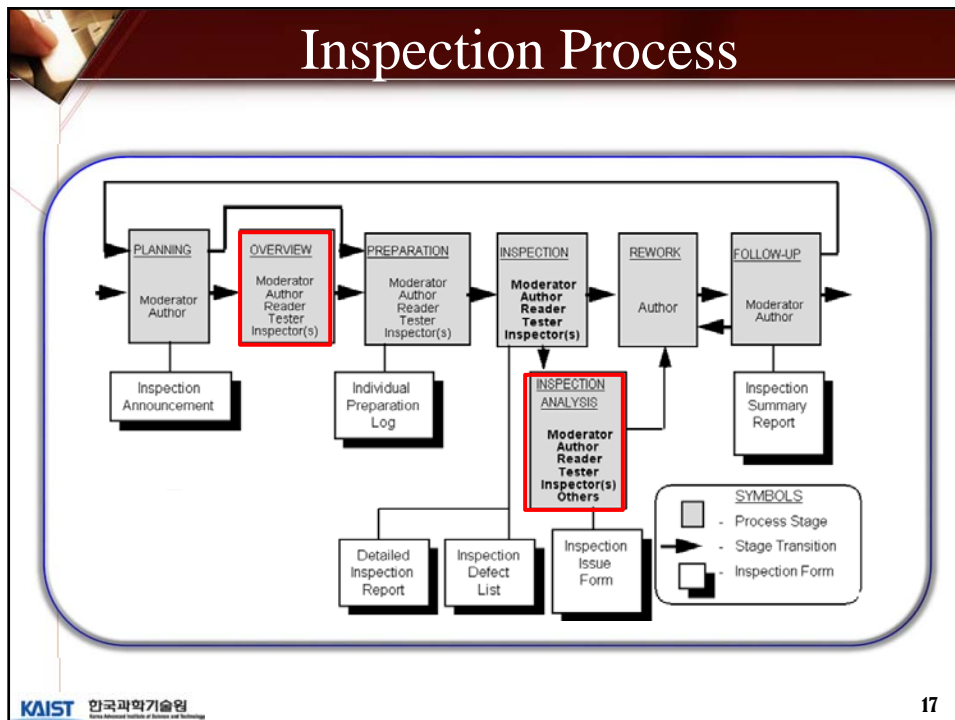
- A hypothetical Example: No Review vs. W/ Review



Software Inspection

- Suggested by Michael Fagan (IBM) in 1976
- Comprised of 7 Operation stages
- Its objectives are to:
 - Find all the defects in the work product that is examined
 - Find all the systemic defects in the process that created defects in the work product.
- It also enabled:
 - Measurement of defects
 - Management of defect rework
 - Removal of systemic defects from the development process





Example: Requirement Checklist

1. Do requirements exhibit distinction between functions and data?
 2. Do requirements define all the information to be displayed to users?
 3. Do requirements address system and user response to error condition?
 4. Is each requirement stated clearly, concisely, and unambiguously?
 5. Is each requirement testable?
 6. Are there ambiguous or implied requirements?
 7. Are there conflicting requirements?
 8. Are there areas not addressed in the Software Requirement Specification (SRS) that need to be?
 9. Are performance requirements (such as response time, data storage requirements) stated?
-

Example: Code Checklist

1. **Data (DA)**
 - ✓Is each variable correctly typed?
 - ✓Is each variable initialized before use?
 - ✓Is the initialization appropriate for the type?
 - ✓Can global variables be made local?
 - ✓Are buffers appropriately sized?
 - ✓Are buffer overflows checked?
 - ✓Is dynamically allocated memory freed?
2. **Interface (IF)**
 - ✓Are appropriate values returned from functions?
 - ✓Do function calls have correct parameter types/values?
 - ✓Are return values tested?
 - ✓Are parameters passed by reference modified correctly?
 - ✓Are parameters passed by value not modified?
3. **Functionality (FN)**
 - ✓Do comparisons use the correct logic?
 - ✓Do loops terminate?
 - ✓Do all loops iterate the correct number of times (no off-by-one errors)?
 - ✓Is behavior correct if a loop is never entered?
 - ✓Is there dead (unreachable) code?
 - ✓Do all switch statements have a default case?
 - ✓Do all switch arms have break statements? If not, is the "fall through" correct?
4. **Input/Output (IO)**
 - ✓Are files opened before use?
 - ✓Are files closed after use?
 - ✓Are buffers flushed at correct times?
 - ✓Are error conditions checked?
5. **Other (OT)**
 - ✓Any defect discovered that does not fall into one of the above categories?

Example: Inspection Record

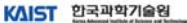
Page ___ of ___ Date __/__/__

Code Inspection Log

Team _____
 Module to be inspected _____

| Role | Name | Prep time | Meeting time |
|------------|-------|-----------|--------------|
| Moderator | _____ | _____ | _____ |
| Author | _____ | _____ | _____ |
| Reader | _____ | _____ | _____ |
| Inspector | _____ | _____ | _____ |
| Total Time | | _____ | _____ |

Total Engr effort _____ # of lines _____
 Number of critical problems _____
 ...


21

Inspection Effectiveness Factors

* The effectiveness of inspection in finding defects that are present

**The envelope of operation
 within which inspections
 find most defects**

- Based upon measured results

Preparation
Rate

Moderator effectiveness

Material meets
Entry Criteria

Domain
Knowledge

Inspection
Rate

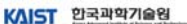
Size of
Inspection Team

Within
2 hours

Phantom Inspector

Language Knowledge

http://www.sdm.de/download/sdm-konf2001/f_7_fagan.pdf


22

Statistical SQA Steps

1. Information about software defects is collected and categorized
2. An attempt is made to trace each defect to its underlying cause
 - e.g.; non-conformance to specification, design error, violation of standards, poor communication with the customer, etc.
3. Using the Pareto principle
 - 80% of the defects can be traced to 20% of all the possible causes (“vital few”)
4. Once the vital few causes have been identified, move to correct the problems that have caused the defects

Reliability & Availability

- Reliability – “the probability of failure-free operation of computer program in a specified environment for a specified time”
 - $MTBF = MTTF + MTTR$
- Availability – “the probability that a program is operating according to requirements as a given point in time”
 - $Availability = [MTTF / (MTTF + MTTR)] * 100\%$
 $= (MTTF / Reliability) * 100\%$

MTBF: Mean Time Between Failure

MTTF: Mean Time To Failure

MTTR: Mean Time To Repair

Classification of SRMs

| Issues | Prediction | Estimation |
|---------------------|---|--|
| Data Reference | - Uses historical project data | - Uses observed data from the current project |
| When to Use in SDLC | - Usually made prior to development or test phase | - Usually made later in life cycle |
| Time Frame | - Predict reliability at some future time | - Estimate reliability at either present or some future time |

Project characteristics (Process, Product, Platform...)

Software Development Life Cycle

```

    graph LR
      Req((Req.)) --> Design((Design))
      Design --> Code((Code))
      Code --> Test((Test))
      Test --> Operation((Operation))
  
```

Faults/Failure Data Collection

Early Reliability Prediction Late Reliability Estimation

KAIST 한국과학기술원

25

Software Safety

“A SQA activity that focus on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail”

- Example: hazards associated w/ a cruise control for an automobile
 - Causes uncontrolled acceleration that cannot be stopped
 - Does not respond to depression of brake pedal
 - Does not engage when switch is activated
 - Slowly loses or gains speed
- Software Reliability vs. Safety
 - Closely Related to on another
 - Software Reliability uses statistical analysis to determine the likelihood that a software failure will occur
 - Software Safety examine the ways in which failures result in conditions that can lead to a mishap

KAIST 한국과학기술원

26

Quality Related Standards

- [ISO/IEC 25012:2008](#) - Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) - Data quality model
- [ISO 9001:2008](#) - Quality management systems – Requirements
- IEEE Std 730-1998 - IEEE Standard for Software Quality Assurance Plans
- IEEE 1633™-2008 - IEEE Recommended Practice on Software Reliability
- IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems
-

Q & A

