Software Engineering and Information System

Lecture 08-01: Software Quality Assurance(SQA)

Md. Al-Hasan

Department of Computer Science & Engineering (CSE)

Bangladesh Army University of Science & Technology (BAUST)

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 6/e* (McGraw-Hill 2005). Slides copyright 2005 by Roger Pressman.

Chapter 26

- **Quality Management**

Software Engineering: A Practitioner's Approach, 6/e
by Roger S. Pressman

Slides copyright © 1996, 2001, 2005 by Roger S. Pressman

Quality Management

- Quality concepts
- Software quality assurance
- Software reviews
- Statistical software quality assurance
- Software reliability, availability, and safety
- SQA plan

(Source: Pressman, R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2005)

Quality Concepts

What is Quality Management

- Also called software quality assurance (SQA)
- Serves as an <u>umbrella activity</u> that is applied throughout the software process
- Involves doing the software development <u>correctly</u> versus doing it over again
- Reduces the amount of <u>rework</u>, which results in lower costs and improved time to market

Quality Defined

- Defined as a characteristic or attribute of something
- Refers to <u>measurable</u> characteristics that we can compare to known standards
- In software, it involves such measures as cyclomatic complexity, cohesion, coupling, function points, and source lines of code

Quality Defined (continued)

- Two kinds of quality are sought out
  - Quality of <u>design</u>
    - The characteristic that designers specify for an item
    - This encompasses requirements, specifications, and the design of the system
  - Quality of <u>conformance</u> (i.e., implementation)
    - The degree to which the design specifications are followed during manufacturing
    - This focuses on how well the implementation follows the design and how well the resulting system meets its requirements
- Quality also can be looked at in terms of user satisfaction

  User satisfaction = compliant product + good quality + delivery within budget and schedule

Quality Control

- Involves a series of inspections, reviews, and tests used throughout the software process
- Ensures that each work product meets the requirements placed on it
- Includes a feedback loop to the process that created the work product
  - This is essential in minimizing the errors produced
- Combines measurement and feedback in order to adjust the process when product specifications are not met
- Requires all work products to have defined, measurable specifications to which practitioners may compare to the output of each process

The Cost of Quality

- Includes all costs incurred in the pursuit of quality or in performing quality-related activities
- Is studied to
  - Provide a baseline for the current cost of quality
  - Identify opportunities for reducing the cost of quality
  - Provide a normalized basis of comparison (which is usually dollars)
- Involves various kinds of quality costs (See next slide)
- Increases dramatically as the activities progress from
  - Prevention Detection Internal failure External failure

"It takes less time to do a thing right than to explain why you did it wrong." Longfellow

Kinds of Quality Costs

- <u>Prevention</u> costs
  - Quality planning, formal technical reviews, test equipment, training
- <u>Appraisal</u> costs
  - Inspections, equipment calibration and maintenance, testing
- <u>Failure</u> costs – subdivided into <u>internal</u> failure costs and <u>external</u> failure costs
  - <u>Internal</u> failure costs
    - Incurred when an error is detected in a product <u>prior to</u> shipment
    - Include rework, repair, and failure mode analysis
  - <u>External</u> failure costs
    - Involves defects found <u>after</u> the product has been shipped
    - Include complaint resolution, product return and replacement, help line support, and warranty work

Software Quality Assurance

Software Quality Defined

Definition: "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software"

(More on next slide)

Software Quality Defined (continued)

- This definition emphasizes three points
  - Software requirements are the foundation from which quality is measured; lack of conformance to requirements is lack of quality
  - Specified standards define a set of development criteria that guide the manner in which software is engineered; if the criteria are not followed, lack of quality will almost surely result
  - A set of implicit requirements often goes unmentioned; if software fails to meet implicit requirements, software quality is suspect
- Software quality is no longer the sole responsibility of the programmer
  - It extends to software engineers, project managers, customers, salespeople, and the SQA group
  - Software engineers apply solid technical methods and measures, conduct formal technical reviews, and perform well-planned software testing

Software Reviews

Purpose of Reviews

- Serve as a filter for the software process
- Are applied at various points during the software process
- Uncover errors that can then be removed
- Purify the software analysis, design, coding, and testing activities
- Catch large classes of errors that escape the originator more than other practitioners
- Include the formal technical review (also called a walkthrough or inspection)
  - Acts as the most effective SQA filter
  - Conducted by software engineers for software engineers
  - Effectively uncovers errors and improves software quality
  - Has been shown to be up to 75% effective in uncovering design flaws (which constitute 50-65% of all errors in software)

Defect Amplification and Removal

- Section: 26.3.2

(Self)

Statistical Software Quality Assurance

Process Steps

1) Collect and categorize information (i.e., causes) about software defects that occur
2) Attempt to trace each defect to its underlying cause (e.g., nonconformance to specifications, design error, violation of standards, poor communication with the customer)
3) Using the Pareto principle (80% of defects can be traced to 20% of all causes), isolate the 20%

A Sample of Possible Causes for Defects

- Incomplete or erroneous specifications
- Misinterpretation of customer communication
- Intentional deviation from specifications
- Violation of programming standards
- Errors in data representation
- Inconsistent component interface
- Errors in design logic
- Incomplete or erroneous testing
- Inaccurate or incomplete documentation
- Errors in programming language translation of design
- Ambiguous or inconsistent human/computer interface