MODULE 5

Software Quality

Badariah Solemon 2010

Objectives:

- To explain the concepts and principles of quality, software quality and software quality management.
- To explain several indicators of a high quality software product.
- To present real life quality dilemmas for even the best SE organizations.
- To describe four broad activities that help a software team achieve high quality software.
- To elaborate the concepts, principles, methods and standards for Software Quality Assurance (SQA)

Software Quality

- In 2005, *ComputerWorld* [Hil05] lamented that
  - "bad software plagues nearly every organization that uses computers, causing lost work hours during computer downtime, lost or corrupted data, missed sales opportunities, high IT support and maintenance costs, and low customer satisfaction.
- A year later, *InfoWorld* [Fos06] wrote about the
  - "the sorry state of software quality" reporting that the quality problem had not gotten any better.
- Today, software quality remains an issue, but who is to blame?
  - Customers blame developers, arguing that sloppy practices lead to low-quality software.
  - Developers blame customers (and other stakeholders), arguing that irrational delivery dates and a continuing stream of changes force them to deliver software before it has been fully validated.

What is quality?

- The *American Heritage Dictionary* defines *quality* as
  - "a characteristic or attribute of something."
- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.
  - User satisfaction = compliant product + good quality + delivery within budget and schedule

What is software quality?

- Software quality can be defined as:
  - *An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.*
- Effective process:
  - establishes the infrastructure that supports any effort at building a high quality software product
  - The management aspects of process create the checks and balances that help avoid project chaos—a key contributor to poor quality.

What is software quality? (cnt'd)

- Useful product:
  - delivers the content, functions, and features that the end-user desires in a reliable, error free way.
  - Satisfies requirements of stakeholders and that are expected of all high quality software.
- Adding value:
  - high quality software provides benefits to producer and user
  - To producer:
    - less maintenance effort, fewer bug fixes, and reduced customer support
  - To user:
    - expedites some business process.
  - End results:
    - Increased revenue, better profitability when an application supports a business process, and/or improved availability of information that is crucial for the business.

Quality Dimensions and Factors

- Can be used as generic quality indicators of a software product.
  - *Garvin Quality Dimensions*
  - *McCall's Quality Factors*
  - *ISO 9126 Quality Factors*
  - *Targeted Factors*

Quality Dimensions and Factors (cnt'd)

Garvin	McCall	ISO 9126	Targeted
1. Performance Quality 2. Feature Quality Reliability 3. Conformance 4. Durability Serviceability 5. Aesthetic 6. Perception	1. Correctness 2. Reliability 3. Efficiency 4. Integrity 5. Usability 6. Maintainability 7. Flexibility 8. Testability 9. Portability 10. Reusability 11. Interoperability	1. Functionality 2. Reliability 3. Usability 4. Efficiency 5. Maintainability 6. Portability	1. Intuitiveness 2. Efficiency 3. Robustness 4. Richness

pg. 401- 405

Quality Dimensions

- David Garvin [Gar87]:
  - **Performance Quality.** Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end-user?
  - **Feature quality.** Does the software provide features that surprise and delight first-time end-users?
  - **Reliability.** Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error free?
  - **Conformance.** Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?

Quality Dimensions

- **Durability.** Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?
- **Serviceability.** Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period. Can support staff acquire all information they need to make changes or correct defects?
- **Aesthetics.** Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious "presence" that are hard to quantify but evident nonetheless.
- **Perception.** In some situations, you have a set of prejudices that will influence your perception of quality.

Quality Dimensions and Factors (cnt'd)

- Efficiency
  - The degree to which the software makes optimal use of software resources.
- Usability
  - The degree to which the software is easy to learn, use, operate, prepare input for and interpret output from.
- Maintainability
  - The ease with which repair maybe made to the software
- Reliability
  - The amount of time that the software is available for use

The Software Quality Dilemmas

- If you produce a software system that has terrible quality, you lose because no one will want to buy it.
- If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway.
- Either you missed the market window, or you simply exhausted all your resources.
- So people in industry try to get to that magical middle ground where the product is **good enough** not to be rejected right away, such as during evaluation, but also not the object of so much perfectionism and so much work that it would take too long or cost too much to complete. [Ven03]

Achieveing Software Quality

- Broad activities that help a software team achieve high quality software:
  1. *Quality assurance (QA) – establishes the infrastructure that supports solid software engineering methods, rational project management, and quality control actions.*
  2. *Quality control (QC) – action that helps ensure each work products meets its quality goals (e.g., Review design models to ensure that they are complete and consistent).*
  3. *Software engineering method – understand the problem to be solved, create a design that conforms to the problems and exhibit characteristics that lead to software that are reliable, efficient, usable, etc.*
  4. *Project management technique –use estimation to verify that delivery dates are achievable, schedule dependencies are understood and conduct risk planning so that problem do not breed chaos.*

Software Quality Assurance (SQA)

- Encompasses:
  1. An SQA process
  2. Specific QA and QC tasks – technical review, audits, multitier testing strategy etc.
  3. Effective SE practice (methods and tools) – risk management
  4. Control of all software work products and changes made to them – change management, security management
  5. A procedure to ensure compliance with standards – IEEE, ISO, CMMI, Six Sigma etc
  6. Measurement and reporting mechanisms – SQA group

Elements of Software Quality Assurance

- **Standards**.

  The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. Standards may be adopted voluntarily by a software engineering organization or imposed by the customer or other stakeholders. The job of SQA is to ensure that standards that have been adopted are followed and that all work products conform to them.

  Reviews and audits. Technical reviews are a quality control activity per-formed by software engineers for software engineers . Their intent is to uncover errors.

  Audits are a type of review performed by SQA personnel with the intent of ensuring that quality guidelines are being followed for software engineering work.

ELEMENTS OF SQA

- **Testing**. Software testing (Chapters 22 through 26) is a quality control function that has one primary goal—to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted so that it has the highest likelihood of achieving its primary goal. Error/defect collection and analysis. The only way to improve is to measure how you're doing. SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them.

  Change management. Change is one of the most disruptive aspects of any software project. If it is not properly managed, change can lead to confusion, and confusion almost always leads to poor quality. SQA ensures that adequate change management practices have been instituted.

ELEMENTS OF SQA

- **Education.** Every software organization wants to improve its software engineering practices. A key contributor to improvement is education of software engineers, their managers, and other stakeholders. The SQA organization takes the lead in software process improvement and is a key proponent and sponsor of educational programs.
  Vendor management. Three categories of software are acquired from external software vendors—shrink-wrapped packages (e.g., Microsoft Office), a tailored shell that provides a basic skeletal structure that is custom tailored to the needs of a purchaser, and contracted software that is custom designed and constructed from specifications provided by the customer organization.
- The job of the SQA organization is to ensure that high-quality software results by suggesting specific quality practices that the vendor should follow (when possible), and incorporating quality mandates as part of any contract with an external vendor.

ELEMENTS OF SQA

- **Security management**. With the increase in cyber crime and new government regulations regarding privacy, every software organization should institute policies that protect data at all levels, establish firewall protection for WebApps, and ensure that software has not been tampered with internally. SQA ensures that appropriate process and technology are used to achieve software security .
  Safety. Because software is almost always a pivotal component of human-rated systems (e.g., automotive or aircraft applications), the impact of hidden defects can be catastrophic. SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.
  Risk management. Although the analysis and mitigation of risk is the concern of software engineers, the SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

SQA Tasks

1. Prepares an SQA plan for a project.
   - The plan identifies:
     - evaluations to be performed
     - audits and reviews to be performed
     - standards that are applicable to the project
     - procedures for error reporting and tracking
     - documents to be produced by the SQA group
     - amount of feedback provided to the software project team
2. Participates in the development of the project's software process description.
   - The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.

SQA Tasks(cnt'd)

3. Reviews software engineering activities to verify compliance with the defined software process.
   - identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
4. Audits designated software work products to verify compliance with those defined as part of the software process.
   - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
   - periodically reports the results of its work to the project manager.
5. Ensures that deviations in software work and work products are documented and handled according to a documented procedure.
6. Records any noncompliance and reports to senior management.
   - Noncompliance items are tracked until they are resolved.

SQA Goals

- Requirements quality. The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- Design quality. Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- Code quality. Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- Quality control effectiveness. A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

Summary:

In this module, students have been introduced to:

- The concepts and principles of quality, software quality and software quality management.
- Several indicators of a high quality software product.
- A real life quality dilemmas for even the best SE organizations.
- Four broad activities that help a software team achieve high quality software.
- The concepts, principles, methods and standards for Software Quality Assurance (SQA)

References and credit

- Contents in the slides are adapted from the book and the slides that accompanied the book by R.S. Pressman, Software Engineering: A Practitioner's Approach, 7th. Edition, McGraw Hill, 2009.