Software Quality Engineering

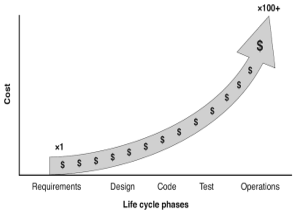
Instructor: Behjat Zuhaira

Errors vs. faults vs. failures

- Software errors
  - Sections of the code that are incorrect grammatically or logically because of a member of the software development team
- Software faults
  - Software errors that cause incorrect functioning
- Software failures
  - Software faults becomes software failures when a user try to run a faulty software section

Defect prevention vs. detection

- Reducing number of defects/errors in the software
- *Defect prevention*: cost effective, eliminates rework
  - Through process improvement
  - Increasing staff knowledge and skill
  - Surveys, training, market analysis
- *Defect detection:* identify and correct the defect as early in the product life cycle as possible (less rework)
  - Reviews, testing



What is Quality?

The word quality has multiple meanings.Two of these meanings dominate the use of the word: 1. Quality consists of those product features which meet the need of customers and thereby provide product satisfaction. 2. Quality consists of freedom from deficiencies. Nevertheless, *… short definition of the word quality [is] … "fitness for use".*

- *Software Quality Assurance*: planned and systematic set of all actions and activities to provide adequate confidence that:
  - Software deliverables (work products) conform to their standards and quality is being built into them
  - Organization's QMS is adequate to meet organization's quality goals and is properly planned, being followed, and is effective.
- *Software Quality Control:* planned and systematic set of all actions and activities needed to monitor and measure software projects, processes, and products to ensure that unwanted variations have not been introduced.

Quality Attributes

- Software engineering approaches ensure that software is built with certain desirable attributes
- The quality attributes set is a way to represent customer quality requirements

McCall's Quality Attributes

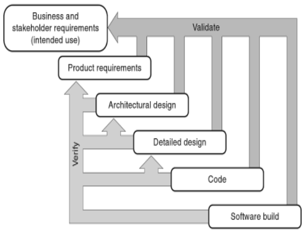
- Operational characteristics
  - Accuracy
  - Reliability
  - Efficiency
  - Integrity
  - Usability
- Ability to be changed
  - Maintainability
  - Flexibility
  - Testability
- Adaptability to new environments
  - Portability/transferability
  - Reusability
  - Interface facility/interoperability

- Efficiency: volume of code and/or computer resources needed for a program to fulfill its function.
  - A software system is efficient if it uses computing resources economically
- Integrity: measures a system's ability to withstand attacks (both accidental and intentional) on its security
  - safeguarding against illegal access to program and data (computer viruses)
- Reliability: probability that the software will operate as expected over a specified time period
  - A program is estimated to have a reliability of 0.96 if, out of 100 executions, it operates correctly 96 times
- Usability: user friendliness
  - The cost/effort to learn and operate a product is a good measure of usability
- Accuracy: The extent to which a program fulfills it specification
  - The equivalence between the software and its specification
  - Accuracy is the function of the features of consistency, completeness and easy interpretation of requirements

- Maintainability: the ease with which a program can be
  - corrected if an error is encountered,
  - adapted if its environment changes, or
  - enhanced if the customer desires a change in requirements
  - maintainability of software incurs cost
    - the dominant cost in the life cycle of a product is running and maintenance costs
- Testability: the cost of program testing for the purpose of safeguarding that specific requirements are met
- Flexibility: the cost and efforts required to modify an operational program
  - Flexibility is present if new requirements can be accommodated
- Interface Facility: Cost of connecting two products with one another
- Portability: the cost and efforts required to transfer a program from one hardware and/or software environment to another
- Reusability: the extent to which a program or part of a program can be reused in other application
  - OO software development methodology

V&V

- **Verification**: *"Process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase"*
  - *Is the software being built right?*
  - Software meets its specified product requirements
  - Adheres to appropriate standards, practices, and conventions
    - □ For example: code is verified against the detailed design and the detailed design against the architectural design
- **Validation**: *"Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled"*
  - *Is the right software being built?*
  - Software meets its intended objectives
  - Matches the user's actual needs



V&V Methods

- Static analysis
  - Evaluates a software work product to assess its quality and identify defects WITHOUT actually executing that work product
    - Examples: reviews, analysis, mathematical proofs
      - □ Reviews: formal design review (FDR), walkthroughs, inspection, desk checks
        - □ **Formal vs. informal reviews**
- Dynamic analysis
  - Evaluating a software component or product by executing it and comparing actual results to expected results
    - Examples: testing, simulations, piloting
      - □ Testing: black box vs. white box, different levels of testing, functional and non-functional testing

Static Analysis

- Evaluates a software work product to assess its quality and identify defects WITHOUT actually executing that work product
  - Example: analysis, reviews, and mathematical proofs
- *Analysis*
  - Types of analysis: hazard analysis, security analysis, risk analysis, requirements allocation and traceability analysis
  - can also be performed using tools such as spell checkers, grammar checkers, compilers, code analyzers
- *Reviews* includes product and process reviews to evaluate the completeness, correctness, consistency, and accuracy
  - Includes entry/exit criteria reviews, quality gates (FDR), peer reviews (desk-checks, walkthroughs, inspections), other forms of managerial and technical reviews

Static Analysis

- *Mathematical proof/proof of correctness*
  - A formal technique used to prove mathematically that a computer program satisfies its specified requirements
  - Mathematical logic deduces that the logic of the design or code is correct

Dynamic Analysis

- Evaluating a software component or product by executing it and comparing actual results to expected results
  - Examples: testing and simulations, piloting

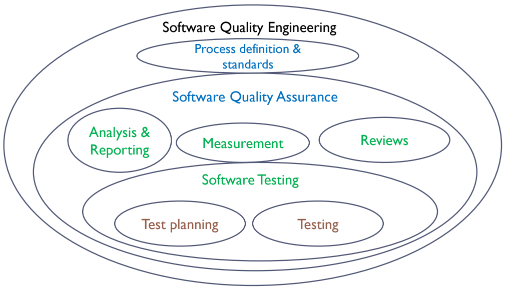
Quality Assurance (QA)

- QA is a system consisting of methods and processes which interact in such a way that software products meet the requirements demanded
  - The system includes planning, estimating and supervision of development activities which are performed independently of the developer
  - Consists of auditing and reporting functions for management
- Goal: provide management with the data necessary to be informed about product quality
- SQE, SQA and software testing are overlapping concepts

Quality Assurance System (QAS)

- A framework for all QA measures and strategies, including
  - Construction and release procedures
  - Allocation of responsibilities for QA
  - Selection of tools for implementation of QA
- Levels of QAS
  - *Company specific level:* overlapping projects
  - *Project specific level:* each project has its own QA team
  - *Phase specific level:* deals with individual phases of the project

Elements of SQA



Software Quality Engineering

Process definition & standards

Software Quality Assurance

Analysis & Reporting

Measurement

Reviews

Software Testing

Test planning

Testing

Elements of Software Quality Assurance (SQA)

- Standards
- Reviews and audits
- Testing
- Error/defect collection and analysis
- Change management
- Education
- Vendor management
- Security management
- Safety
- Risk management

Standards

- Formal mandatory requirements developed and used to prescribe consistent approaches to development (SEI: 2006)
- Used as a basis of comparison
- ISO and IEEE are two well-renowned designated standards body
- Guidelines vs. Standards vs. Regulations vs. Models
  - Standards define requirements; considered mandatory
  - Guides define suggested practices, advice, methods, or procedures that are considered good practice but are not mandatory
  - Regulations are rules/laws established by a legislative/regulatory body; non-compliance results in penalties (e.g. fine, jail-time etc.)
  - A Model is an abstract representation of an item/process from a particular point of view; no unnecessary details; not mandatory (CMMI, lifecycle models, etc.)

Why standards?

- Easy flow of events
- Reduced efforts
- Uniform method of analysis/evaluation
- Increase professionalism of a discipline
- Easy access to good practices as defined by the experienced practitioners

Reviews and audits

- Technical reviews are a quality control activity performed 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.
  - For example, an audit of the review process might be conducted to ensure that reviews are being performed in a manner that will lead to the highest likelihood of uncovering errors.

Testing

- Software testing 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
  - efficiently conducted so that it has the highest likelihood of achieving its primary goal

Error collection and analysis

- 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

- The most disruptive aspects of any software project
- SQA ensures that adequate change management practices have been instituted

Education

- A key contributor to improvement is education of software engineers, their managers, and other stakeholders
- Educational programs

Vendor management

- ensure that high-quality software results by suggesting specific quality practices that the vendor should follow

Safety

- Assessing the impact of software failure and for initiating those steps required to reduce risk

Security management

- SQA ensures that appropriate process and technology are used to achieve software securely

Risk management

- ensures that risk management activities are properly conducted
- risk-related contingency plans have been established

SQA Activities / role of SQA group

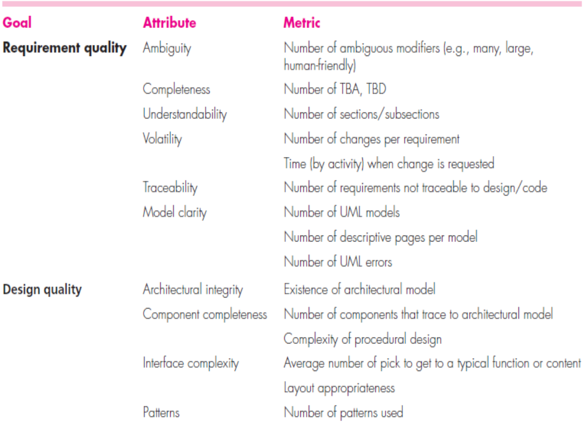
- Preparing a SQA plan for a project
- Participating in the development of the project's software process description
  - Selecting a process for the work
  - Reviewing process description for compliance with organizational policies, internal software standards, external standards (e.g. ISO 9001), etc.
- Reviewing SE activities to verify compliance with the defined software process
  - Identification of documents
  - Identification of deviations from the process
  - Verification of corrections
- Auditing designated software work products to verify compliance with those defined as part of the software process
  - Reviewing selected work products
  - Identifying, documenting and tracking deviations
  - Verifying corrections
  - Periodic reporting

Contd.

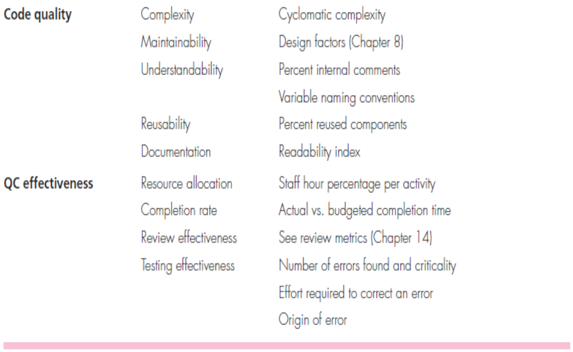
- Ensures that deviations in software work and work products are documented and handled according to documented procedure
  - Deviations may occur in project plan, process description, standards or deliverables
- Records any noncompliance and reports to senior management
  - Noncompliance items are tracked, until resolved
- Other:
  - Coordinating control and management of change
  - Helping to collect and analyze software metrics

SQA Goals

- The SQA activities described in the preceding section are performed to achieve a set of pragmatic 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.



Goal	Attribute	Metric
Requirement quality	Ambiguity	Number of ambiguous modifiers (e.g., many, large, human-friendly)
	Completeness	Number of TBA, TBD
	Understandability	Number of sections/subsections
	Volatility	Number of changes per requirement
		Time (by activity) when change is requested
	Traceability	Number of requirements not traceable to design/code
	Model clarity	Number of UML models
		Number of descriptive pages per model
		Number of UML errors
Design quality	Architectural integrity	Existence of architectural model
	Component completeness	Number of components that trace to architectural model
		Complexity of procedural design
	Interface complexity	Average number of pick to get to a typical function or content
		Layout appropriateness
	Patterns	Number of patterns used



Code quality	Complexity	Cyclomatic complexity
	Maintainability	Design factors (Chapter 8)
	Understandability	Percent internal comments
		Variable naming conventions
	Reusability	Percent reused components
	Documentation	Readability index
QC effectiveness	Resource allocation	Staff hour percentage per activity
	Completion rate	Actual vs. budgeted completion time
	Review effectiveness	See review metrics (Chapter 14)
	Testing effectiveness	Number of errors found and criticality
		Effort required to correct an error
		Origin of error