13. Software Quality Assurance and Quality Management

Abdus Sattar
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University
Email: abdus.cse@diu.edu.bd

Daffodil International University

Discussion Topics

- Software Quality Assurance
- What are SQA, SQP, SQC, and SQM?
- Elements of Software Quality Assurance
- SQA Tasks, Goals, And Metrics
- Quality Management and Software Development
- QA Vs QC
- Reviews and Inspections
- An Inspection Checklist

Software Quality Assurance

- **Software Quality Assurance** (SQA) is a set of activities for ensuring quality in software engineering processes. It ensures that developed software meets and complies with the defined or standardized quality specifications.
- SQA is an ongoing process within the Software Development Life Cycle (SDLC) that routinely checks the developed software to ensure it meets the desired quality measures.

What are SQA, SQP, SQC, and SQM?

- **Software Quality Assurance** – establishment of network of organizational procedures and standards leading to high-quality software
- **Software Quality Planning** – selection of appropriate procedures and standards from this framework and adaptation of these to specific software project
- **Software Quality Control** – definition and performing of processes that ensure that project quality procedures and standards are being followed by the software development team
- **Software Quality Metrics** – collecting and analyzing quality data to predict and control quality of the software product being developed

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.
- **Reviews and audits.** Technical reviews are a quality control activity performed by software engineers for software engineers.
- **Testing.** Software testing is a quality control function that has one primary goal—to find errors.
- **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.
- **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.

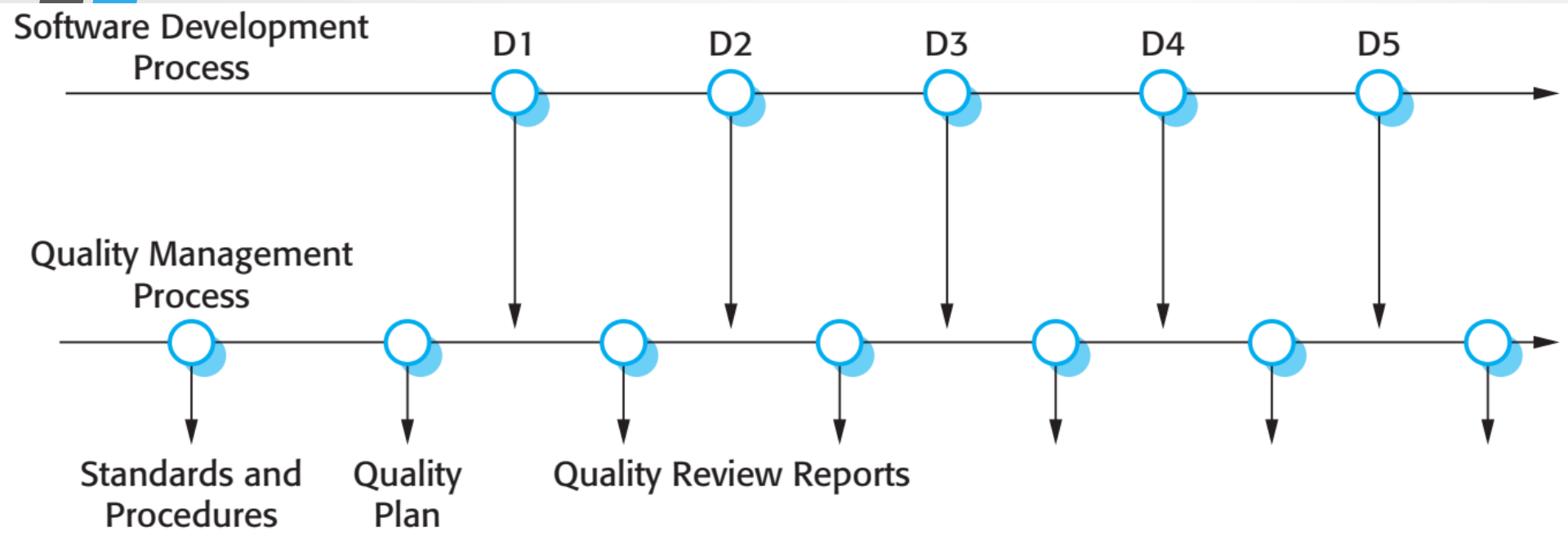
Elements of Software Quality Assurance(Cont…)

- **Vendor management.** Three categories of software are acquired from external software vendors—*shrink-wrapped packages* (e.g., Microsoft *Office*), 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.
- **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.** SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.
- **Risk management.** Risk management activities are properly conducted and that risk-related contingency plans have been established

SQA Tasks, Goals, And Metrics

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

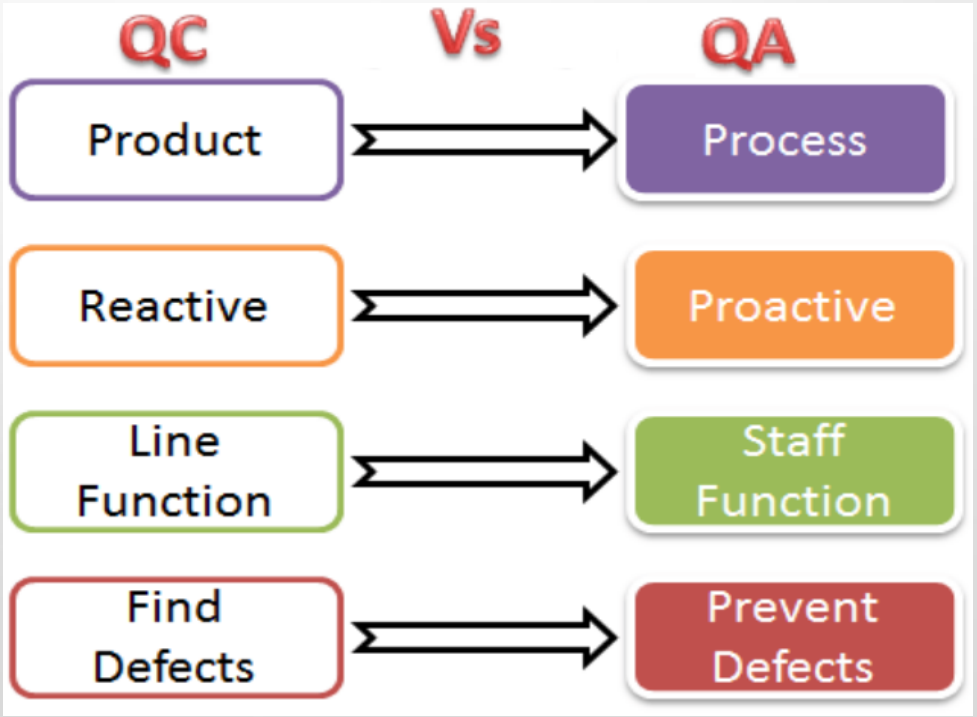
Quality Management and Software Development



- **Quality management** provides an independent check on the software development process.
- **The quality management** process checks the project deliverables to ensure that they are consistent with organizational standards and goals.
- QA team should be independent from the development team so that they can take an objective view of the software. This allows them to report on software quality without being influenced by software development issues.

Quality Management

- The terms ‘quality assurance’ and ‘quality control’ are widely used in manufacturing industry.
- **Quality Assurance (QA)** is the definition of processes and standards that should lead to high-quality products and the introduction of quality processes into the manufacturing process.
- **Quality Control(QC)** is the application of these quality processes to weed out products that are not of the required level of quality.



Reviews and Inspections

- **Reviews and inspections** are QA activities that check the quality of project deliverables. This involves examining the software, its documentation and records of the process to discover errors and omissions and to see if quality standards have been followed.
- **During a review**, a group of people examine the software and its associated documentation, looking for potential problems and non-conformance with standards. The review team makes informed judgments about the level of quality of a system or project deliverable. Project managers may then use these assessments to make planning decisions and allocate resources to the development process.
- **Program inspections** are 'peer reviews' where team members collaborate to find bugs in the program that is being developed.
- **Program inspections** involve team members from different backgrounds who make a careful, line-by-line review of the program source code. They look for defects and problems and describe these at an inspection meeting. Defects may be logical errors, anomalies in the code that might indicate an erroneous condition or features that have been omitted from the code. The review team examines the design models or the program code in detail and highlights anomalies and problems for repair.
- **During an inspection**, a checklist of common programming errors is often used to focus the search for bugs. This checklist may be based on examples from books or from knowledge of defects that are common in a particular application domain.

An Inspection Checklist

Fault class	Inspection check
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?
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?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?
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?
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?

❑ References:

1. **Software Engineering A practitioner's Approach**

   by Roger S. Pressman, 7th edition, McGraw Hill, 2010.

2. **Software Engineering by Ian Sommerville,**

   9th edition, Addison-Wesley, 2011