Benghazi University
Faculty of Information Technology
Software Engineering Department

Software Quality

Part4
(Quality Management)

Instructor: Salah Abdelsattar

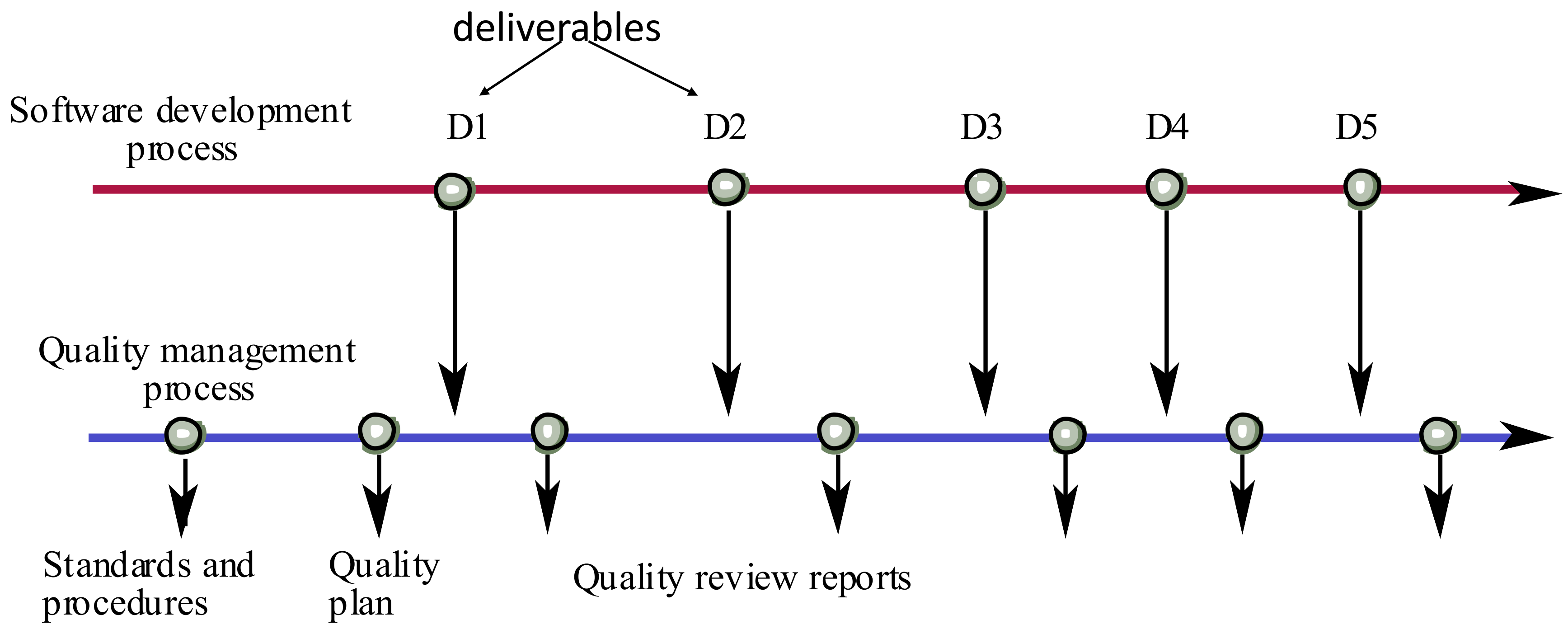
Software Quality Management (SQM)

Goals: Ensure that the required level of quality is achieved in software products, namely, that defined standards and procedures are followed

SQM activities:

- Quality Assurance (QA)
  - Quality Assurance is all the planned and systematic activities implemented within the quality system to provide confidence that the project will satisfy the relevant quality standards.
- Quality Control (QC)
  - Quality Control involves monitoring specific project results to determine if they comply with relevant quality standards, and identifying ways to eliminate causes of unsatisfactory results.
- Quality Planning (QP)
  - Quality planning involves identifying which quality standards are relevant to the project and determining how to satisfy them.

Quality management and software development



Quality Control Vs. Quality Assurance

- Quality control is about work product, quality assurance is about work process.

- Quality control activities are work product oriented. They measure the product, identify deficiencies, and suggest improvements. The direct result of these activities are changes to the product. These can range from single-line code changes to completely reworking a product. It is an *error-removal* technique.

- Quality assurance activities are work process oriented. They measure the process, identify deficiencies, and suggest improvements. The direct result of these activities are changes to the process. These changes can range from better compliance with the process to entirely new processes. The output of quality control activities is often the input to quality assurance activities.

Software Quality Assurance (SQA)

An overall activity that is applied throughout the software engineering process. It is an *error-prevention technique* that focuses on the process of systems and software development**.**

- A set of established frameworks (procedures & standard) used by an organization to achieve high quality software
- Essential activity for any business that produces products to be used by others
  - It need to be planned and systematic
  - It does not just happen

SQA is often thought of as a software testing activity - **WRONG**

- *If quality isn't part of a product prior to testing, it won't be part of the product after testing is completed.*
- *SQA must be part of Software Engineering from the beginning.*
- SQA is applied at every stage of the software process
- Quality standard is selected to apply to the software process.

Software Quality Assurance (SQA)

The goals of SQA

- to improve software quality by monitoring both the process and the product.
- to ensure compliance with all local standards for SE.
- to ensure that any product defect, process variance, or standards non-compliance is noted and fixed.

Scenario

You are hired to manage the software engineering of a new product estimated at 25,000 lines of code, involving three or four people working for about 18 months.

Consider the following:

1. What should be in place before the project starts?
2. What quality checkpoints should you have?
3. What measures will you collect to determine the level of quality of the work being performed?

Principle of SQA

1. Set the standard and quality attributes that a software product must meet
   - The goal to achieve
2. Measure the quality of software product
   - There is a way to determine how well the product conforms to the standards and quality attributes
3. Track the values of the quality attributes
   - It is possible to assess how well we are doing
4. Use the information of software quality to improve the quality of future software product
   - There is a feedback into the software development process

Activities of SQA

- Application of Technical Methods
- Conduct of Formal Technical Reviews
- Software Testing
- Enforcement of Standards
- Control of Change
- Measurement
- Record keeping and Reporting

Application of Technical Methods

- Select appropriate tools and methods to capture system requirements, analyze system, design, implementation and testing

Why use methods and tools

- To ensure high quality is achieved
- Analyst achieve high quality specifications
- Designer develop high quality designs
- Ability to measure quality in specifications and designs

Conduct of Formal Technical Reviews

- FTR is an activity to assess quality
- A stylized meeting conducted by technical staff with the sole purpose of uncovering quality problems
- Found to be effective in uncovering defects in software
- All major software items should be subjected to technical review

Software Testing

- Testing is a process of executing a program with the intent of finding errors, fixing them and to prove system correctness.
- Testing is conducted based on the developed test cases
  - Capture actual output
  - Compare actual output with expected output
    - Actual == Expected : Test case succeed
    - Actual != Expected : Test case failed
- All test cases results must be recorded

Enforcement of Standards

- Formal standards and procedures varies from company to company
- Could be dictated by customer, regulations, or self-imposed
- If formal (written) standards exist, they must be followed to ensure quality
- Assessment of compliance is done through FTR or audit

Control of Change

- An activity executed throughout the system life cycle to control change of products and life cycle artifacts
- Items that need control of changes
  - Plans
  - Specification
  - Procedures
  - Audit
  - Report
  - Support
- A software library need to be constructed that store, manage and track these items

Control of Change

- Every change has potential for introducing errors or creating side effects that propagate errors
- Change control process ensures quality
- Formalizing requests for change, evaluating nature of change, and controlling the impact of change.

Measurement

- Measurement : Numeric derivation for some attribute of a software product / process
  - Compare the value with standard apply
  - Often expressed in $$$ and days
  - Integral part to any engineering discipline
- Software metrics must be collected to track quality
- Also to assess the impact of methodological and procedural changes on improved software quality

Record keeping and Reporting

- Historical record for the project that keeps

  - results of reviews,
  - audits,
  - change control,
  - testing,
  - other SQA activities

- Provides procedures for collection and dissemination of SQA information

- Dissemination to development staff based on need-to-know basis

Quality Control

- Quality control is defined as the processes and methods used to monitor work and observe whether requirements are met. It focuses on reviews and removal of defects before shipment of products.

- This involves checking the software development process to ensure that procedures and standards are being followed.

- Quality control consists of well-defined checks on a product that are specified in the product quality plan.

- For software products, quality control typically includes specification reviews, inspections of code and documents, and checks for user deliverables.

Quality control is usually performed using two methods.

- Reviews of documents such as requirements documents and design documents.

- Testing of code and modules (Next Part, Part5)

Reviews

- Reviews are:
  - A meeting conducted by technical people for technical people
  - A technical assessment of a work product created during the software engineering process
  - A software quality assurance mechanism

- Any work product (including documents) should be reviewed.

- Conducting timely reviews of all work products can often eliminate 80% of the defects before any testing is conducted.

Formal Technical Reviews (FTR)

- The objectives of FTR are:
  - To uncover errors in functions, logic, or implementation for any representation of the software.
  - To verify that the software under review meets its requirements.
  - To ensure that the software has been represented according to predefined standards.
  - To achieve software that is developed in a uniform manner.
  - To make projects more manageable.
- Review meeting's constraints:
  - 3-5 people involved in a review
  - advanced preparation (no more than 2 hours for each person)
  - the duration of the review meeting should be less than 2 hours
  - focus on a specific part of a software product
- People involved in a review meeting:
  - producer, review leader, 2 or 3 reviewers (one of them is recorder)

The FTR meeting

- Producer informs the project leader that the product is complete and that a review is required
- The project leader forms a review team and appoints a review leader
- The review leader evaluate the product for readiness, generates copies of the product material, distributes to the reviewers, and schedules a review meeting
- Each reviewer reviews the product. He becomes familiar with the product and makes notes of concerns
- Review leader, all reviewers, and producer attend the meeting
- One of the reviewers take the role of recorder
- During the meeting, the producer walks through the product, explaining the material, while the reviewers raise issue based on their preparation. If an error is discovered, then it is recorded by the recorder.

Outcome of FTR meeting

- The attendees of the meeting decide whether to:
  - Accept the product without further modification
  - Accept the product provisionally. Minor errors have been encountered. These must be fixed but no additional review will be needed
  - Reject the product due to sever errors. Once the errors are fixed, another review should be conducted
- At the end of an FTR, a review summary report should be produced. It should answer the following
  - What was reviewed?
  - Who was involved in the review?
  - What were the findings and conclusions?

Review Checklists

- Checklists can be used to assess products that are derived as part of software development.

- The checklists are not intended to be comprehensive, but rather to provide a point of departure for each review.

Reviews of Software Requirements Analysis

- Reviews for software requirements analysis focus on traceability to system requirements and consistency and correctness of the analysis model.

- A number of FTRs are conducted for the requirements of a large system and may be also followed by reviews and evaluation of prototypes as well as customer meetings.

Reviews of Software Requirements Analysis

- The following topics are considered during FTRs for analysis:
  - Is information domain analysis complete, consistent and accurate?
  - Is problem partitioning complete?
  - Are external and internal interfaces properly defined?
  - Does the data model properly reflect data objects, their attributes and relationships.
  - Are all requirements traceable to system level?
  - Has prototyping been conducted for the user/customer?
  - Is performance achievable within the constraints imposed by other system elements?
  - Are requirements consistent with schedule, resources and budget?
  - Are validation criteria complete?

Reviews of Software Design

- Reviews for software design focus on data design, architectural design and procedural design.
- In general, two types of design reviews are conducted:
  - The preliminary design review assesses the translation of requirements to the design of data and architecture.
  - The second review, often called a design walkthrough, concentrates on the procedural correctness of algorithms as they are implemented within program modules.

Reviews of Software Design

- The following checklists are useful for preliminary design review:
  - Are software requirements reflected in the software architecture?
  - Is effective modularity achieved? Are modules functionally independent?
  - Are interfaces defined for modules and external system elements?
  - Is the data structure consistent with information domain?
  - Is data structure consistent with software requirements?
  - Has maintainability been considered?
  - Have quality factors been explicitly assessed?
  - Does the algorithm accomplishes desired function?
  - Is the algorithm logically correct?

Reviews of Software Design

- Is the interface consistent with architectural design?
- Is the logical complexity reasonable?
- Has error handling been specified?
- Are local data structures properly defined?
- Are structured programming constructs used throughout?
- Is design detail amenable to implementation language?
- Which are used: operating system or language dependent features?
- Has maintainability been considered?

Reviews of Coding

- Errors can be introduced as the design is translated into a programming language.
- A code walkthrough can be an effective means for uncovering these translation errors.

Reviews of Coding

- The following checklist assumes that a design walkthrough has been conducted and that algorithm correctness has been established as part of the design FTR.
  - Has the design properly been translated into code? [The results of the procedural design should be available during this review.]
  - Are there misspellings and typos?
  - Has proper use of language conventions been made?
  - Is there compliance with coding standards for language style, comments, module prologue?
  - Are there incorrect or ambiguous comments?
  - Are data types and data declaration proper?
  - Are physical constants correct?
  - Have all items on the design walkthrough checklist been re-applied (as required)?

Reviews of Testing

- Software testing is a quality assurance activity in its own right.
- The completeness and effectiveness of testing can be dramatically improved by critically assessing any test plans and procedures that have been created.

Reviews of Testing

- The following checklists are useful for test plan walkthrough:
  - Have major test phases properly been identified and sequenced?
  - Has traceability to validation criteria/requirements been established as part of software requirements analysis?
  - Are major functions demonstrated early?
  - Is the test plan consistent with overall project plan?
  - Has a test schedule been explicitly defined?
  - Are test resources and tools identified and available?
  - Has a test record keeping mechanism been established?

Reviews of Testing

- Have both white and black box tests been specified?
- Have all independent logic paths been tested?
- Have test cases been identified and listed with expected results?
- Is error-handling to be tested?
- Have all boundary values been tested?
- Are timing and performance to be tested?
- Has acceptable variation from expected results been specified?

Reviews of Maintenance

- The review checklists for software development are equally valid for the software maintenance phase. In addition to all of the questions posed in the checklists, the following special considerations should be kept in mind:
  - Have side effects associated with change been considered?
  - Has the request for change been documented, evaluated and approved?
  - Has the change, once made, been documented and reported to interested parties?
  - Have appropriate FTRs been conducted?
  - Has a final acceptance review been conducted to ensure that all software has been properly updated, tested and replaced?

Metrics Derived from Reviews

- Inspection time per page of documentation
- Inspection time per KLOC or FP
- Inspection effort per KLOC or FP
- Errors uncovered per reviewer hour
- Errors uncovered per preparation hour
- Errors uncovered per SE task (e.g., design)
- Number of minor errors (e.g., typos)
- Number of major errors (e.g., nonconformance to requirements)
- Number of errors found during preparation

Software Quality planning

The quality plan

- Provides a road map for instituting software quality assurance in an organization.
- Developed by the SQA group to serve as a template for SQA activities that are instituted for each software project in an organization
- Structured as follows:
  - The purpose and scope of the plan
  - A description of all software engineering work products that fall within the purview of SQA
  - All applicable standards and practices that are applied during the software process
  - SQA actions and tasks (including reviews and audits) and their placement throughout the software process
  - The tools and methods that support SQA actions and tasks
  - Methods for assembling, safeguarding, and maintaining all SQA-related records
  - Organizational roles and responsibilities relative to product quality

The SQA group

Who is involved in quality assurance activities?

Software engineers, project managers, customers, sale people, SQA group

Engineers involved in the quality assurance work:

- apply technical methods and measures
- conduct formal technical review
- perform well-planned software testing

The SQA group's role -> serves as the customer's in-house representative
assist the software engineering team in achieving high-quality

The SQA group's responsibility:

- quality assurance planning oversight, record keeping, analysis and reporting

The SQA group's tasks:

- Prepare a SQA plan for a project
- Participate in the development of the project's software process description
- Review engineering activities to verify compliance with the defined process
- Audits designated software work products to verify compliance with the defined process
- Ensure the deviations in software work and products according to a documented procedure
- Records any noncompliance and reports to senior management