Chapter 24 Notes

Software Quality Management for SW systems has 3 principal concerns (2 at project level):

1. Organization Level
   1. Concerned with establishing a framework of organizational processes and standards that will lead to high-quality software. This mean that the quality management team should take responsibility for defining the software development processes to be used and standards that should apply to the software and related documentation, including the system requirements, design, and code
2. Project Level
   1. Quality Management involves the application of specific quality processes, checking that these planned processes have been followed, and ensuring that the project outputs are conformant with the standards that are applicable to that project
   2. Also concerned with establishing a quality plan for a project. The quality plan should set out the quality goals for the project and define what processes and standards are to be used

Quality Assurance (QA) = the definition of processes and standards that should lead to high-quality products and the introduction of quality processes into the manufacturing process. It also includes verification and validation.

Quality Control (QC) = the application of these quality processes to weed out products that are not of the required level of quality

QA teams are responsible for managing the release testing process (i.e. testing the software before it is released to customers).

Quality Management provides an independent check on the SW development process. It checks for project deliverables to ensure that they are consistent with organizational standards and goals.

Quality Planning is the process of developing a quality plan for a project. It should set out the desired software qualities and describe how these are to be assessed. It therefore defines what “high-quality” software actually means for a system.

Quality Plan Outline:

* Product Introduction
  + Description of the products, its intended market, and the quality expectations for the product
* Product Plans
  + Critical release dates and responsibilities for the product, along with plans for distribution and product servicing.
* Process Descriptions
  + The development and service processes and standards that should be used for product development and management
* Quality Goals
  + Quality goals and plans for the product, including an identification and justification of critical product quality attributes
* Risks and Risk Management
  + The key risks that might affect product quality and the actions to be taken to address these risks

Formalized Quality Managements is particularly important for teams that are developing large, long-lifetime system that takes several years to develop.

Quality documentation is a record of what has been done by each subgroup in the project.

* It helps people check that important tasks have not been forgotten or that one group has not made incorrect assumptions about what other groups have done.
* Is also a means of communication over the lifetime of a system.
* It allows the groups responsible for system evolution to trace the tests can checks that have been implemented by the development team.

For smaller system, quality management is still important but is more informal.

* The key thing for the small systems is to develop a quality culture and ensure that all team members have a positive approach to software quality.

**Software Quality (Section 24.1)**

* Quality is based on conformance with a detail product specification and a tolerance (tolerance though is not applicable to SW systems)
* Reasons to have an objective conclusion about whether or not a software system meets its specification:
  + It is difficult to write complete and unambiguous software specifications. Software developers and customers may interpret the requirements in different ways and thus may not agree whether or not the software conforms to its specification
  + Specifications usually integrate requirements from several classes of stakeholders. These are inevitably a compromise and may not include the requirements of all stakeholder groups. The excluded stakeholder groups may then perceive the system as a poor quality system, even though it implements the agreed requirements
  + It is impossible to measure certain quality characteristics (e.g. maintainability) directly and so they cannot be specified in an unambiguous way.
* Because of the above reasons, assessment of software quality is a subjective process where the quality management team has to use their judgment to decide if an acceptable level of quality has been achieved. The quality management team has to consider whether or not the software is fir for its purpose. This involves answering the following questions about the system’s characteristics:
  + Have programming and documentation standards been followed in the development process?
  + Has the software been properly tested?
  + Is the software sufficiently dependable to be put into use?
  + Is the performance of the software acceptable for normal use?
  + Is the software usable?
  + Is the software well structured and understandable?
* There is a general assumption in software quality management that the system will be tested against its requirements.
  + The judgment on whether or not it delivered the required functionality should be based on the results of these tests. Therefore the QA team should review the tests that have been developed and examine the test records to check that testing has been properly carried out.
* Subjective quality of a software system is based on its non-functional characteristics.
  + Non-functional system quality attributes
    - Safety, Understandability, Portability, Security, Testability, Usability, Reliability, Adaptability, Reusability, Resilience, Modularity, Efficiency, Robustness, Complexity, and Learnability
  + The quality plan should define the most important quality attributes for the SW since not all of the attributes can be satisfied
* Process-based approach to achieving product quality
  + Define Process
  + Develop Product
  + Assess Product Quality
    - If quality OK, continue
    - Else, Improve Process and then go back to Developing Product
  + Standardize the Process

**Software Standards (Section 24.2)**

* An important part of QA is the definition or selection of standards that should apply to the software development process or software product. As part of this QA process, tools and methods to support the use of these standards may also be chosen.
* SW standards are important for 3 reasons
  + Standards capture wisdom that is of value to the organization. They are based on knowledge about the best or most appropriate practice for the company. It helps the company reuse experience and avoid previous mistakes
  + Standards provide a framework for defining what ‘quality’ means in a particular setting.
  + Standards assist continuity when work carried out by one person is taken up and continued by another. Standards ensure that all engineers within an organization adopt the same practices. Consequently, the learning effort required when starting new work is reduced.
* 2 Related Types of SWE Standards that may be defined and used in quality management
  + Product Standards
    - These apply to the software product
    - Include document standards, documentation standards, and coding standards
  + Process Standards
    - These define the processes that should be followed during SW development
    - Encapsulate good development practice
    - Include definitions of specification, design and validation processes, process support tools, and a description of the documents that should be written during these processes.
    - Also should include the definition of processes that check that product standards have been followed
* Engineers find standards to be overprecriptive and not really relevant to the technical activity of SW development. To minimize dissatisfaction and to encourage buy-in to standards, quality managers who set the standards should therefore do the following:
  + Involve SW engineers in the selection of product standards
  + Review and modify standards regularly to reflect changing technologies
  + Provide SW tools to support standards
* **The ISO 9001 Standards Framework (Section 24.2.1)**
  + It is a framework for developing SW standards. It sets out general quality principles, describes quality processes in general, and lays out the organizational standards and procedures that should be defined. These should be documented in an organizational quality manual.
  + ISO 9001 Core Processes
    - Product Delivery Processes
      * Business Acquisition
      * Design and Development
      * Test
      * Production and Delivery
      * Service and Support
    - Supporting Processes
      * Business Management
      * Supplier Management
      * Inventory Management
      * Configuration Management
  + For an organization to conform to this standard it must document how its processes relate to these core processes and it must also define and maintain records that demonstrate that the defined organizational process have been followed.
  + ISO 9001 and Quality Management relationships
    - *ISO 9001 Quality Models*
      * instantiated as *Organization Quality Manual*
    - *Organization Quality Manual*
      * Documents the *Organization Quality Process*
      * Is used to develop the *Project 1, Project 2, … Project N Quality Plans*
    - *Project Quality Management*
      * *Organization Quality Process* is instantiated as this
      * *Project 1, Project 2, … Project N Quality Plans* support this
  + Agile Methods (Which avoid documentation and focus on the code being developed) have little in common with the formal quality processes that are discussed in ISO 9001.
    - Agile development is fundamentally opposed to what they see as the bureaucratic overhead of standards conformance.

**Reviews and Inspections (Section 24.3)**

* Are QA activities that check the quality of project deliverables.
  + This involves examining the SW, its documentation and records of the process to discover errors and omissions and to see if quality standards have been followed.
* During review, a group of people examine the SW and its associated documentation, looking for potential problems and non-conformance with standards. The group 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.
* Reviews are based on documents that have been produced during the development process, as well as SW specifications, designs, code, process models, test plans, configuration management procedures, process standards, and user manuals.
* Reviews also help discover problems and omissions in the SW or project documentation in addition to checking conformance to standards
* The purpose of reviews and inspections is to improve quality.
* **The Review Process (Section 24.3.1)**
  + Review process is structured into 3 phases:
    - Pre-review activities
      * Are preparatory activities that are essential for the review to be effective.
      * Are concerned with review planning and review preparation
      * Planning -> involves setting up a review team, arranging a time and place for the review, and distributing the documents to be reviewed.
      * Review Preparation -> Team may meet to get an overview of the SW to be reviewed. The review team reviews the software/documents and relevant standards. They work independently to find errors, omissions, and departures from standards.
    - Review Meeting
      * An author of a document or program being reviews should “walk through” the document with the review team.
    - Post-Review Activities
      * Issues and problems raised during the review must be addressed. This may involve fixing SW bugs, refactoring SW so that ir conforms to quality standards, or rewriting documents.
  + The review process relies on all development team members being collocated and available for a team meeting.
  + Review Process in different types of development
    - In Agile, the process is usually informal
      * Quality reviews can slow down the pace of SW development and they are best used within a plan-driven process -> is impractical in Agile methods that focus on development
    - In Scrum, there is a review meeting after each sprint
    - In XP, pair programming ensures that code is constantly being examined and review by another team member.
* **Program Inspection (Section 24.3.2)**
  + These are peer reviews where team member collaborate to find bugs in the program that is being developed.
  + Inspections may be part of the validation and verification processes.
  + They complement testing and do not require execution.
  + During an inspection, a checklist of common programming errors is often used to focus the search for bugs
  + Agile processes rarely use formal inspection or review processes. Rather, they rely on team member cooperating to check each other’s code, and informal guidelines, such as “check before check-in”
  + Example Checklist
    - Data Faults
      * Are all variables initialized before values are used?
      * Have all constants been named?
      * Is there a possibility of buffer overflow?
    - Control Faults
      * For each conditional statement, is the condition correct?
      * Is each loop certain to terminate?
      * In case statements, are all possible cases account for?
    - I/O faults
      * Are all input variables used?
      * Can unexpected inputs cause corruption?
    - Interface Faults
      * Do all function calls have the correct number of parameters?
      * Are the parameters in the right order?
    - Storage Management Faults
      * If dynamic storage is used, has space been allocated correctly?
      * Is space explicitly de-allocated after it is no longer required?
    - Exception Management Faults
      * Have all possible error conditions been taken into account?

**Software Measurement and Metrics (Section 24.4)**

* SW Measurement is concerned with deriving a numeric value or profile for an attribute of a software component, system, or process. By comparing these values to each other and to the standards that apply across an organization, you may be able to draw conclusions about the quality of software, or assess the effectiveness of software processes, tools, and methods.
* The long-term goal of this is to use measurement in place of reviews to make judgments about software quality. Using measurements, a system could ideally be assessed using a range of metrics and a value for the quality of the system could be inferred.
* A SW metric is a characteristic of a SW system, system documentation, or development process that can be objectively measured. (e.g., lines of code).
* SW Metrics can be either:
  + Control/Process Metrics
    - Support process management
    - Are associated with software processes
    - Ex. Average effort and time required to repair reported defects
  + Predictor/Product Metrics
    - Are associated with the SW itself and are also known as “Product Metrics
    - Ex. The average length of identifiers in a program
* Managers use Process/Control metrics to decide if process changes should be made and use Predictor/Product Metrics to help estimate the effort required to make software changes.
* 2 Ways in which Measurements of a SW system may be used:
  + To assign a value to system quality attributes like maintainability
  + To identify the system components whose quality is substandard
* Unfortunately, it is difficult to make direct measurements of many of the software quality attributes
* There are many relationships between external (e.g., maintainability) and internal (e.g., total lines of code, number of error message, cyclomatic complexity, etc.) attributes
  + If the measure of the internal attribute is to be a useful predictor of the external SW characteristics, 3 condition must hold:
    - The internal attribute must be measure objectively
    - A relationship must exist between the attribute that can be measured and the external quality attribute that is of interest
    - This relationship between the external and internal attributes must be understood, validated, and expressed in terms of a formula or model.
* There is little info publicly available about the current use of systematic SW measurement in industry. Many companies do collect info about their SW. However, it is not clear if they then use these measurements systematically to compare software products and processes or assess the impact of changes to software processes and tolls. There are several reasons why this is difficult:
  + It is impossible to quantify the return on investment of introducing an organizational metrics program
  + There are no standards for software metrics or standardized processes for measurement and analysis. Many companies are reluctant to introduce measurement programs until such standards and supporting tools are available
  + In many companies, SW processes are not standardized and are poorly defined and controlled -> there is too much process variability within the same company for measurements to be used in a meaningful way
  + Much of the research of SW measurement and metrics has focused on code-based metrics and plan-driven development processes.
  + Introducing measurement adds additional overhead to processes. This contradicts the aims of agile methods, which recommend the elimination of process activities that are not directly related to program development.
* **Product Metrics (Section 24.4.1)**
  + These are predictor metrics that are used to measure internal attributes of a software system.
    - Examples are system size or number of methods in a program
  + Product Metrics fall into 2 categories
    - Dynamic Metrics
      * Are collected by measurements made of a program in execution.
      * Can be collected during system testing or after the system has been deployed.
      * An example might be the number of bug reports
      * Are used to assess the Efficiency and Reliability of a program
      * There is usually a clear relationship between these metrics and software quality characteristics
    - Static Metrics
      * Collected by measurements made of representations of the system, such as the design, program, or documentation.
      * Examples are the code size and average length of identifiers used.
      * Help assess the Complexity, Understandability, and Maintainability of a system
      * Have an indirect relationship with quality attributes
      * Examples
        + Fan-in/Fan-out

Fan-in is a measure of the number of functions that call another function X. Fan-out is the number of functions that are called by function X. High value for fan-in means X is tightly coupled to the rest of the system and a high value for fan-out means that X is complex.

* + - * + Length of Code

The larger the program, the more complex and error-prone it is

* + - * + Cyclomatic Complexity

Measure of the control complexity of a program

* + - * + Length of Identifiers

Measure of the average length of identifiers in a program. The longer they are, the more understandable a program is likely to be.

* + - * + Depth of conditional nesting

Measure of the depth of nesting if-statements in a program. Deeper nests = less understandable code

* + - * + Fog Index

Measure of the average length of words and sentences in a document. Higher this is, the lower understandability is

* + Object Oriented Metrics
    - Weighted Methods per Class
      * Number of methods in each class, weighted by their complexity
      * Larger = more complex object classes
    - Depth of Inheritance Tree
      * Number of discrete levels in the inheritance tree
      * Deeper = more complex design
    - Number of Children
      * Measure of the number of immediate subclasses in a class.
      * Higher values = higher reusability
    - Copling between Object Classes
      * Classes are coupled when methods in one class use methods or instance variables defined in a different class
      * Measure of how much coupling exists
      * High value = higher dependability (harder to maintain)
    - Response for a Class
      * Measure of the number of methods that could potentially be executed in aresponse to a message received by an object of that class
      * Higher value = more complex a class
    - Lack of Cohesion in Methods
      * Is calculated by considering pairs of methods in a class. LCOM is the difference between the number of method pairs without shared attributes and the number of method pairs with shared attributes
* **SW Component Analysis (Section 24.4.2)**
  + Component Measurement Process
    - Choose measurements to be made
    - Select components to be assessed
      * You may only need to assess a subset of the entire system in order to make an overall assessment of the system quality or you can just inspect subsystems by themselves.
    - Measure component characteristics
      * Selected components are measured and the associated metric values computed.
    - Identify anomalous measurements
      * Compare measurements with each other and to previous measurements that have been recorded in a measurement database. Look for unusually high or low values for each metric, as these suggest that there could be problems with the component exhibiting these values.
    - Analyze anomalous components
      * Examine components with anomalous values for the chosen metrics and determine if the quality of the component has been compromised.
* **Measurement Ambiguity (Section 24.4.3.)**
  + When you collect quantitative data about SW and SW processes, you have to analyze that data to understand its meaning. It is easy to misinterpret data and to make inferences that are incorrect. You cannot simply look at the data on its own – you must also consider the context where the data is collected.
  + Reasons why users might make change requests:
    - The software is not good enough and does not do what customer want it to do. They therefore request changes to deliver the functionality that they require
    - Alternatively, the software may be very good and so it is widely and heavily used. Change requests may be generated because there are many SW users who creatively think of new things that could be done with the software.