Introduction to software Quality and Assurance

By,
Swathi Bhat

Introduction

- **Software is**: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
- **Software engineering** is a branch of computer science that involves the systematic design, development, testing, maintenance, and documentation of software in order to create high-quality and reliable computer systems.

- **Software testing** is a critical phase in the software development life cycle (SDLC) that ensures the quality and reliability of software applications.
- Software testing is the process of evaluating a software application to identify any difference between expected and actual results.
- To ensure that the software meets specified requirements, works as intended, and is free from defects or errors.

- **Key Goals of Software Testing**
- **Identifying Defects:** Discovering and documenting defects or errors in the software.
- **Ensuring Functionality:** Verifying that the software functions according to specified requirements.
- **Improving Quality:** Enhancing the overall quality and reliability of the software product.
- **Validating Requirements:** Confirming that the software meets customer expectations and requirements.

Software quality

- IEEE definition
- Software quality is:
- 1. The degree to which a system, component, or process meets specified requirements.
- 2. The degree to which a system, component, or process meets customer or user needs or expectations.

- Quality means conformance to requirements”
- “(1) Quality consists of those product features which meet the needs of customers and thereby provide product satisfaction.
- (2) Quality consists of freedom from deficiencies”

- Software quality – Pressman's definition
- Roger S. Pressman is a software engineer, author, and consultant who is well-known for his contributions to the field of software engineering.
- Software quality is defined as: Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

- Specific functional requirements (functionality/ performance), which refer mainly to the outputs of the software system.
- ■ The software quality standards mentioned in the contract.
- ■ Good Software Engineering Practices (GSEP) (well-established and recognized best practices in software engineering that are generally followed by professionals), reflecting state-of-the-art professional practices, to be met by the developer even though not explicitly mentioned in the contract.
- In effect, Pressman's definition provides operative directions for testing the degree to which the requirements are met.

- **Software quality assurance** – The IEEE definition
- Software quality assurance is:
- 1. A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
- 2. A set of activities designed to evaluate the process by which the products are developed or manufactured.

- This definition may be characterized in the following:
  - Plan and implement systematically.
- SQA is based on planning and the application of a variety of actions that are integrated into all the stages of the software development process.
- This is done in order to substantiate (prove) the client's confidence that the software product will meet all the technical requirements.

Software quality assurance vs. software quality control

- Two phrases are constantly repeated within the context of software quality: “Quality control” and “quality assurance”.
- Quality control is defined as “a set of activities designed to evaluate the quality of a developed or manufactured product” (IEEE, 1991).
- quality control inspection and other activities take place as the development or manufacturing of the product is completed yet before the product is shipped to the client.
- Quality control activities are only a part of the total range of quality assurance activities.

- The main objective of **quality assurance** is to minimize the cost of guaranteeing quality by a variety of activities performed throughout the development and manufacturing processes/stages.
- These activities prevent the causes of errors, and detect and correct them early in the development process.
- As a result, quality assurance activities substantially reduce the rate of products that do not qualify for shipment and, at the same time, reduce the costs of guaranteeing quality in most cases.

The software quality challenge

- The software quality challenge is an issue in the field of software development.
- It encompasses various aspects related to the effectiveness, efficiency, reliability, and maintainability of software systems.
- Addressing these challenges is crucial to delivering high-quality software products.

- Addressing the software quality challenge requires a holistic approach (involves the use of various testing techniques during the software testing process), incorporating robust development practices, effective testing strategies, and continuous improvement processes.
- Regular training, and a focus on user feedback are also essential elements in achieving and maintaining software quality.

- Chapter Outline:
- The uniqueness of Software Quality Assurance
- The environments for which SQA methods are developed

- Identify the unique characteristics of software
- Recognize the characteristics of the environment where professional software development and maintenance take place.
- Explain the main environmental difficulties faced by software development and maintenance teams

The fundamental differences between software products (including firmware) and other products are caused by the **higher product complexity**, by the **invisibility of software** and by the **nature of the product development and production process**.

Firmware is a specific type (or subset) of software .It is a form of microcode or program embedded into hardware devices to help them operate effectively

Uniqueness of SQA

- **1. Product complexity:** can be measured by the number of operational modes(start-up, normal operation, shutdown, product transitions, maintenance, and similar activities)the product permits.
- An industrial product, even an advanced machine, does not allow for more than a few thousand modes of operation, created by the combinations of its different machine settings.
- Looking at a typical software package one can find millions of software operation possibilities.
- Assuring that the a lot of operational possibilities is correctly defined and developed is a major challenge to the software industry

- 2. **Product visibility:** Whereas the industrial products are visible, software products are invisible.
- Most of the defects in an industrial product can be detected during the manufacturing process.
- Moreover the absence of a part in an industrial product is, as a rule, highly visible (imagine a door missing from your new car).
- However, defects in software products (whether stored on diskettes or CDs) are invisible, as is the fact that parts of a software package may be absent from the beginning

3.

- **Product Development and Production Progress**
  - The only phase when defects can be detected is the development phase.

- the possibility of detecting defects in an industrial product may arise:
- (a) **Product development.** In this phase the designers and quality assurance (QA) staff check and test the product prototype, in order to detect its defects.
- (b) **Product production planning.** During this phase the production process and tools are designed and prepared.
- This phase thus provides additional opportunities to inspect the product, which may reveal defects that "escaped" the reviews and tests conducted during the development phase.

- **(c) Manufacturing.**
- At this phase QA procedures are applied to detect failures of products themselves.
- Defects in the product detected in the first period of manufacturing can usually be corrected by a change in the product's design or materials or in the production tools, in a way that eliminates such defects in products manufactured in the future.

- In comparison to industrial products, software products do not benefit from the opportunities for detection of defects at all three phases of the production process.
- The only phase when defects can be detected is the development phase.

- **(a) Product development.** During this phase, efforts of the development teams and software quality assurance professionals are directed toward detecting inherent product defects.
- At the end of this phase an approved prototype, ready for reproduction, becomes available.
- **(b) Product production planning.** This phase is not required for the software production process, as the manufacturing of software copies and printing of software manuals are conducted automatically.

- **(c) Manufacturing.** As mentioned previously, the manufacturing of software is limited to copying the product and printing copies of the software manuals.
- Consequently, expectations for detecting defects are quite limited during this phase.

Table 1.1: Factors affecting defect detection in software products vs. other industrial products

Characteristic	Software products	Other industrial products
Complexity	Usually, very complex product allowing for very large number of operational options	Degree of complexity much lower, allowing at most a few thousand operational options
Visibility of product	Invisible product, impossible to detect defects or omissions by sight (e.g. of a diskette or CD storing the software)	Visible product, allowing effective detection of defects by sight
Nature of development and production process	Opportunities to detect defects arise in only one phase, namely product development	Opportunities to detect defects arise in all phases of development and production: - Product development - Product production planning - Manufacturing

Frame 1.1 **The uniqueness of the software development process**

- **High complexity**, as compared to other industrial products
- **Invisibility of the product**
- **Opportunities to detect defects ("bugs")** are limited to the product development phase

SQA Environment

The software developed by many individuals and in different situations fulfills a variety of needs:

- **Software development fulfills variety of needs**
  - Students as part of their education
  - Software amateurs as a hobby
  - Professionals in engineering, economics, management, and other fields to assist them in their work
  - Software development professionals as a professional career objective (for organizations)

Characteristics of SQA Environment

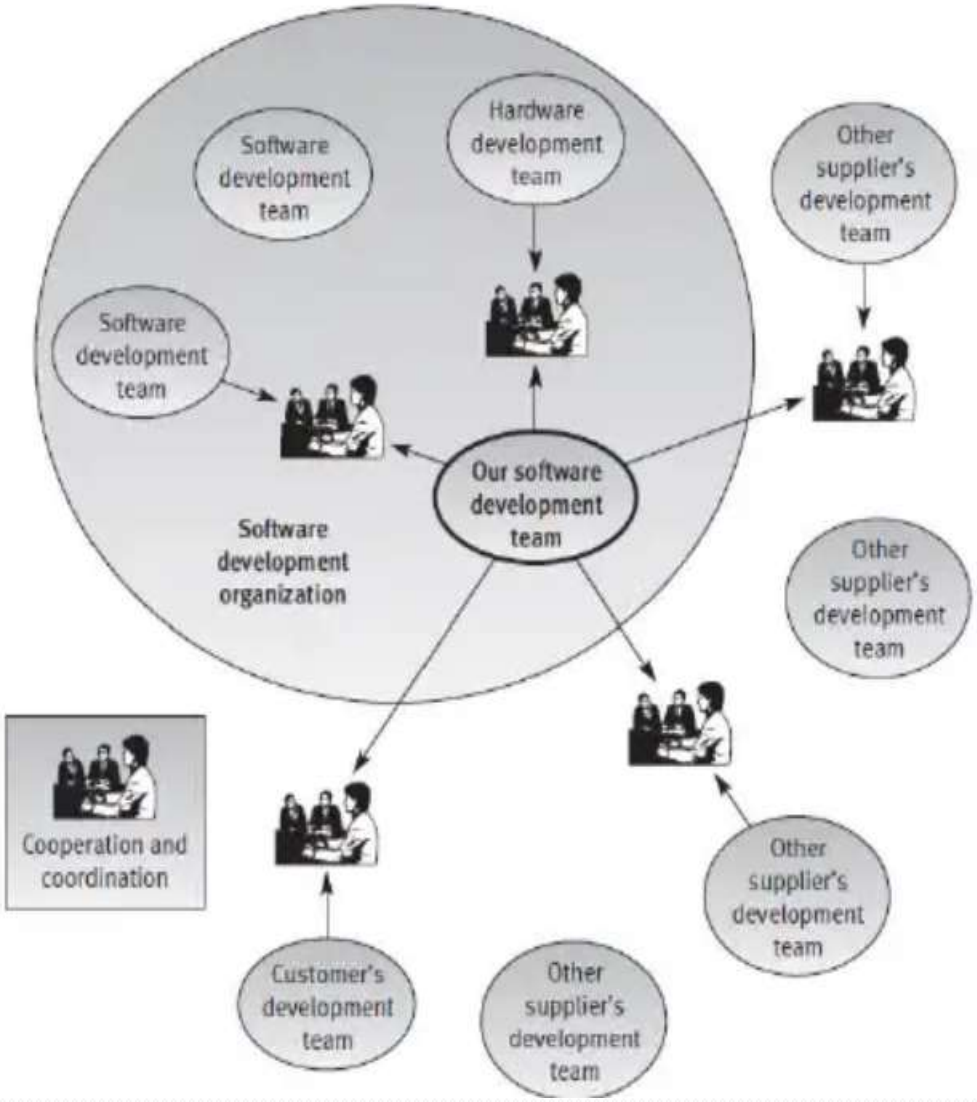
- 1. **Contractual conditions.**
- As a result of the commitments and conditions defined in the contract between the software developer and the customer, the activities of software development and maintenance need to cope with:
- ■ A defined list of functional requirements that the developed software and its maintenance need to fulfill.
  ■ The project budget.
- ■ The project timetable.

- 2. **Subjection to customer–supplier relationship.** Throughout the process of software development and maintenance, activities are under the oversight of the customer.
- The project team has to cooperate continuously with the customer: to consider his request for changes, to discuss his criticisms about the various aspects of the project, and to get his approval for changes initiated by the development team.
- Such relationships do not usually exist when software is developed by non-software professionals.

- **(3) Required teamwork.** Three factors usually motivate the establishment of a project team rather than assigning the project to one professional:
- ■ Timetable requirements. In other words, the workload undertaken during the project period requires the participation of more than one person if the project is to be completed on time.
- ■ The need for a variety of specializations in order to carry out the project.
- ■ The wish to benefit from professional mutual support and review for the enhancement of project quality.

- (4) **Cooperation and coordination with other software teams.**
- The carrying out of projects, especially large-scale projects, by more than one team is a very common event in the software industry.
- In these cases, cooperation may be required with:
- ■ Other software development teams in the same organization.
- ■ Hardware development teams in the same organization.
- ■ Software and hardware development teams of other suppliers.
- ■ Customer software and hardware development teams that take part in the project's development.

4. Cooperation and coordination of software



- (5) **Interfaces with other software systems.**
- Nowadays, most software systems include interfaces with other software packages.
- These interfaces allow data in electronic form to flow between the software systems.
- One can identify the following main types of interfaces:
- ■ Input interfaces, where other software systems transmit data to your software system.
- ■ Output interfaces, where your software system transmits processed data to other software systems.

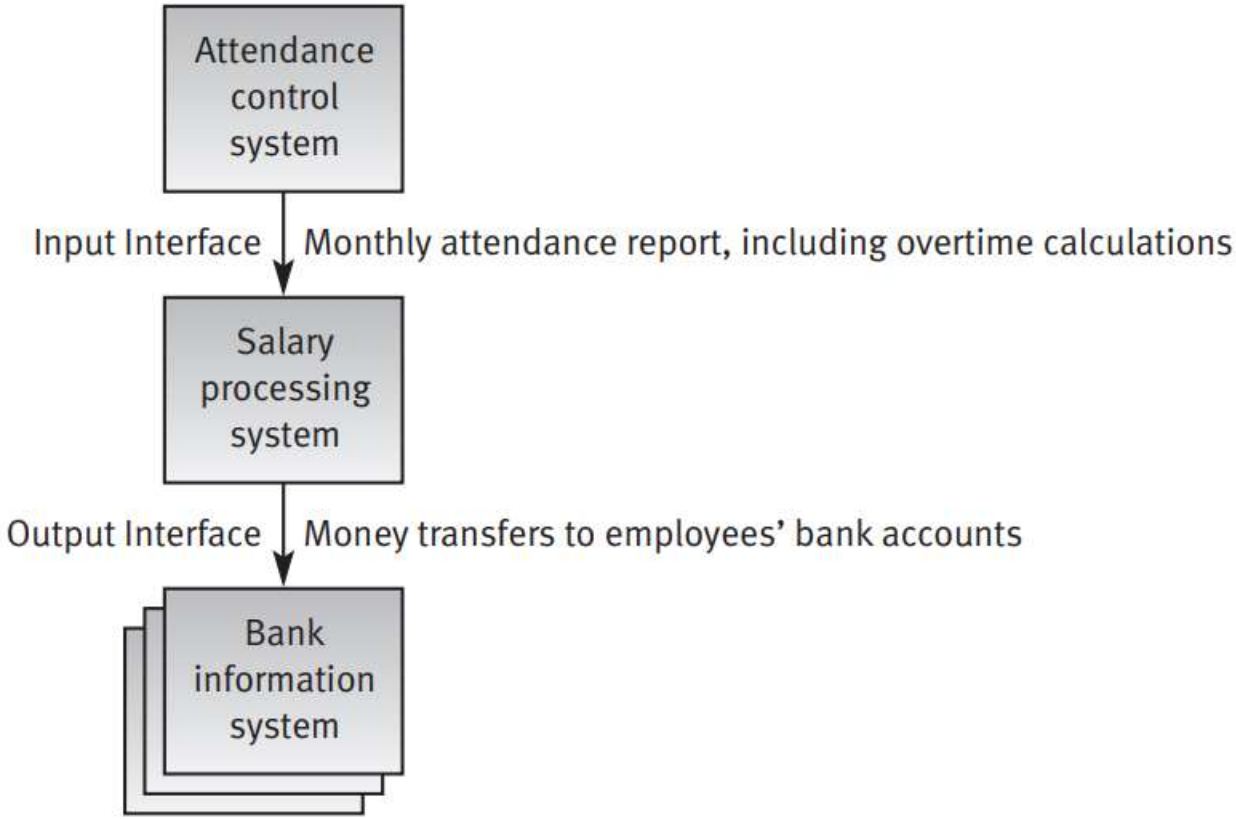


Figure 1.2: The salary software system – an example of software interfaces

- **(6) The need to continue carrying out a project despite team member changes.**
- It is quite common for team members to leave the team during the project development period, whether owing to promotions to higher level jobs, a switch in employers, transfers to another city, and so forth.
- The team leader then has to replace the departing team member either by another employee or by a newly recruited employee.
- No matter how much effort is invested in training the new team member, “the show must go on”, which means that the original project contract timetable will not change.

- **(7) The need to continue carrying out software maintenance for an extended period.**
- Customers who develop or purchase a software system expect to continue utilizing it for a long period, usually for 5–10 years.
- During the service period, the need for maintenance will eventually arise.
- In most cases, the developer is required to supply these services directly.
- Internal “customers”, in cases where the software has been developed in-house, share the same expectation regarding the software maintenance during the service period of the software system.

Frame 1.2 **Summary of the main characteristics of SQA environment**

1. Being contracted
2. Subjection to customer–supplier relationship
3. Requirement for teamwork
4. Need for cooperation and coordination with other development teams
5. Need for interfaces with other software systems
6. Need to continue carrying out a project while the team changes
7. Need to continue maintaining the software system for years

Software quality factors

Quality Factor

- The various *attributes* of software and its use and maintenance, as defined in software requirements documents ,can be classified into content groups called quality factors.

•Software Quality attributes help to measure the quality of the software from different angles. (Usability (User-friendly)
•Efficiency. ...
•Flexibility. ...
•Reliability. ...
•Maintainability. ...
•Portability.
)

The need for comprehensive software quality requirements

- There is a need for a comprehensive definition of requirements that will cover all attributes of software and aspects of the use of software, including *usability aspects*, reusability aspects, *maintainability aspects*, and so forth in order to assure the *full satisfaction of the users*.

- This statement emphasizes the importance of creating a detailed and inclusive set of requirements for software development.
- It suggests that a comprehensive (detailed)definition of requirements is necessary to address all aspects of software and its usage.
- The mentioned attributes include usability (how easy it is to use the software), reusability (the ability to use components of the software in different contexts), maintainability (how easily the software can be maintained and updated), and other factors that contribute to the overall satisfaction of users.

- By considering a wide range of aspects, the aim is to develop software that not only functions well but is also user-friendly, can be reused in various scenarios, and is easy to maintain and update over time.
- This approach is meant to enhance the overall satisfaction and effectiveness of the software for its users.

Classifications of software requirements into software quality factors

- Several models of software quality factors and their categorization in factor categories have been suggested over the years.
- The classic model of software quality factors, suggested by McCall, consists of 11 factors (McCall et al., 1977).
- The McCall model you're referring to was proposed by John McCall and his colleagues in their paper titled "Factors in Software Quality" published in 1977.
- John D. McCall was a computer scientist known for his work in software engineering.
- McCall's factor model classifies all software requirements into 11 software quality factors.
- The 11 factors are grouped into three categories – product operation, product revision and product transition – as follows:

Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product operation factors**:

    Correctness, Reliability, Efficiency, Integrity, Usability.

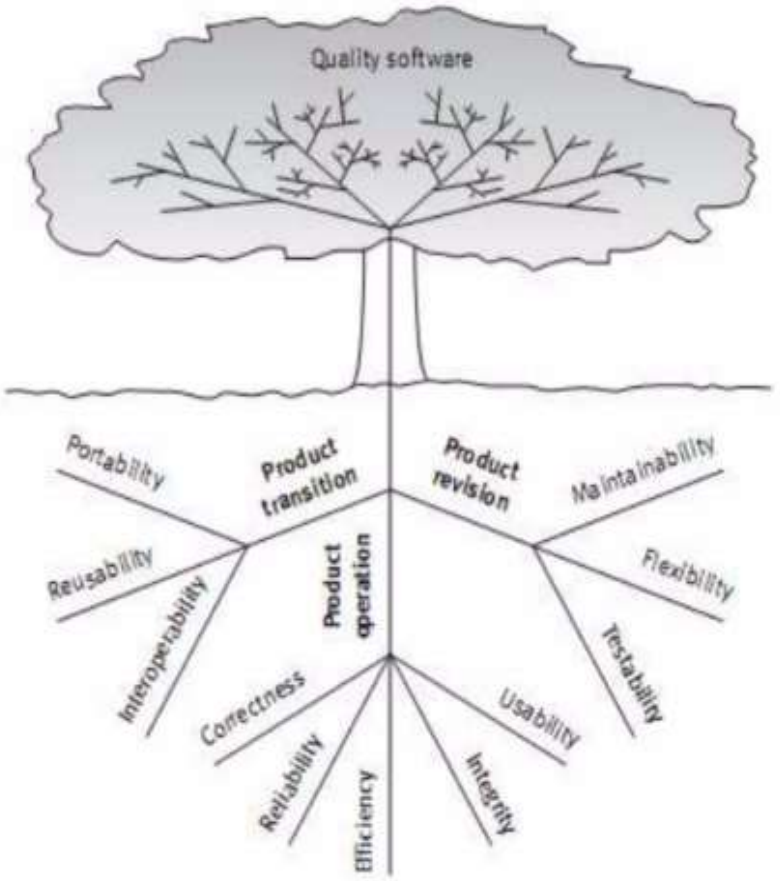
  - **Product revision factors**:

    Maintainability, Flexibility, Testability.

  - **Product transition factors**:

    Portability, Reusability, Interoperability.

Figure 3.1: McCall's factor model tree



1. Product operation software quality factors

- What is the meaning of reliability?
- the quality of being able to be trusted or believed because of working or behaving well.

Correctness

- Correctness requirements are defined in a list of the software system's required outputs.
  - The output mission
  - Accuracy of output
  - Completeness of the output information
  - Up-to-date of the information
  - Availability of the information
  - The standard for coding and documenting of software system

- The output mission (e.g., sales invoice printout, and red alarms when temperature rises above 250°F).

- The term "output mission" refers to the specific outcomes or results that a system or process aims to achieve.
- In the given examples:
- The desired outcome is the generation and printing of a sales invoice.
- Visual or audible alarms, possibly in the color red, triggered when the temperature sensor detects a temperature exceeding 250°F. This could be part of a larger system that monitors and controls temperature in a given environment.

- The required accuracy of those outputs that can be adversely affected by inaccurate data or inaccurate calculations.
- The completeness of the output information, which can be adversely affected by incomplete data.
- The up-to-dateness of the information (defined as the time between the event and its consideration by the software system).
- The availability of the information (the reaction time, defined as the time needed to obtain the requested information or as the requested reaction time of the firmware installed in a computerized apparatus). (machine)
- The standards for coding and documenting the software system.

The software and its documentation are required to comply(meet) with the client's guidelines.

Integrity

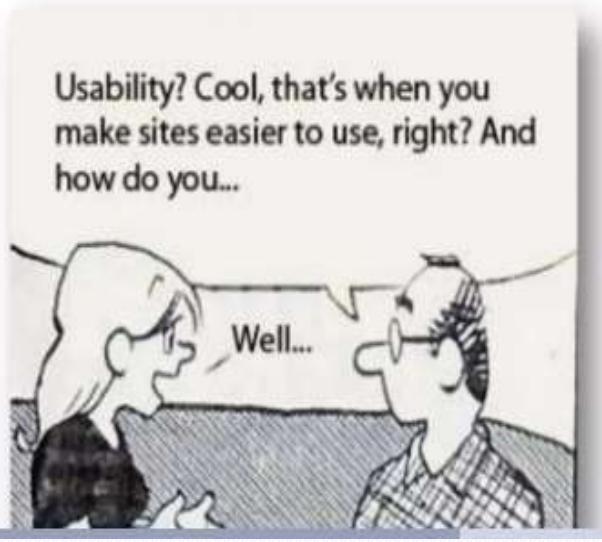
- Integrity requirements *deal with the software system security*, that is, requirements to prevent access to unauthorized persons, to distinguish between the majority of personnel allowed to see the information ("read permit") and a limited group who will be allowed to add and change data ("write permit"), and so forth.

Efficiency

- Efficiency requirements *deal with the hardware resources* needed to perform all the functions of the software system in conformance to all other requirements. The main hardware resources to be considered are the computer's processing capabilities (measured in MIPS – million instructions per second ,etc.)

Usability

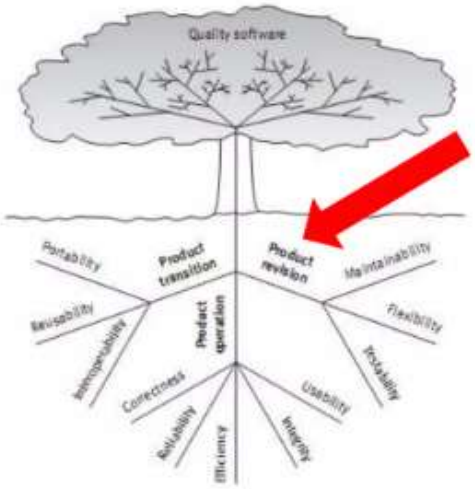
- Usability requirements *deal with the scope of staff resources* needed to train a new employee and to operate the software system.



Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product revision factors**:

    Maintainability, Flexibility, Testability.



Maintainability

- Maintainability requirements determine the efforts that will be needed by users and maintenance personnel to identify the reasons for software failures, *to correct the failures*, and to *verify the success of the corrections*.

  This factor's requirements refer to the modular structure of software, the internal program documentation, and the programmer's manual, among other items.

A modular structure involves dividing the software into smaller, independent units or modules. Each module focuses on a specific functionality or aspect of the system. The idea is to create a well-organized, interconnected set of modules that can be easily understood and modified.

Flexibility

- The capabilities and efforts required to *support adaptive maintenance activities* are covered by the flexibility requirements. These include the resources(i.e. in man-days) required to adapt a software package to a variety of customers of the same trade, of various extents of activities, of different ranges of products and so on.

  This factor's requirements also support perfective maintenance activities.

  such as changes and additions to the software in order to improve its service

- Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system.
- The term "resources in man-days" suggests the effort required for adaptive maintenance is measured in terms of person-days, indicating the number of workdays it would take for a person to complete the necessary adaptations.

Testability

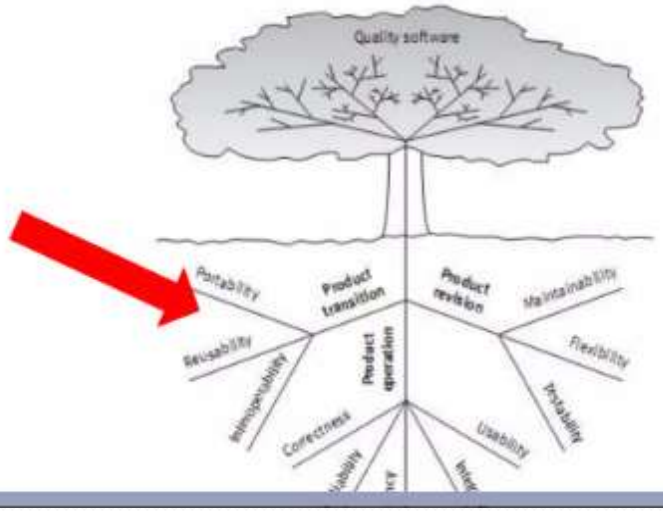
- Testability requirements *deal with the testing of an information system as well as with its operation*. Testability requirements for the ease of testing are related to special features in the programs that help the tester, for instance by providing predefined intermediate results and log files.

- Providing Predefined Intermediate Results:
- This suggests that the software has built-in mechanisms to generate expected results at various stages of execution. These predefined results act as benchmarks against which actual results can be compared during testing.
- Log Files:
- Log files are records of events that occur during the execution of a program. In the context of testing, having well-structured and detailed log files can be immensely helpful. Testers can use these logs to trace the flow of execution, identify issues, and understand the behavior of the software during testing.

Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product transition factors**:

    Portability, Reusability, Interoperability.



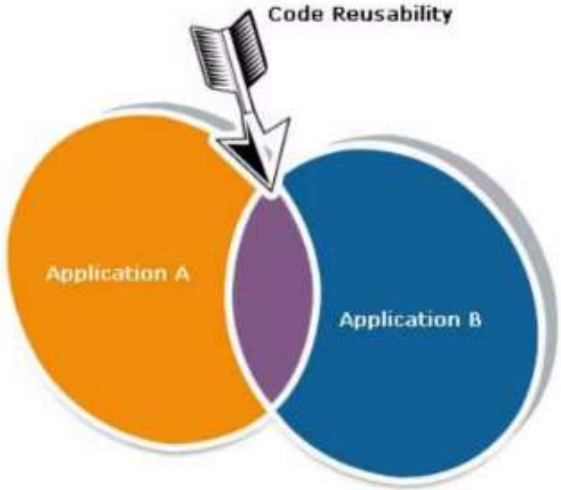
Portability

- Portability requirements tend to the *adaptation of a software system to other environments* consisting of different hardware, different operating systems.

- These requirements make it possible to continue using the same basic software in diverse situations or to use it simultaneously in diverse hardware and operating systems situations.

Reusability

- Reusability requirements deal with the use of software modules originally designed for one project in a new software project currently being developed.



Interoperability

- Interoperability requirements *focus on creating interfaces with other software systems* or with other equipment firmware (for example, the firmware of the production machinery and testing equipment interfaces with the production control software).

Interoperability in software engineering refers to the ability of different software systems or components to work together, exchange information, and use each other's functionality seamlessly.
It is a measure of how well diverse systems can interact and communicate with one another, typically in a way that allows them to function together effectively

Management and its role in software quality assurance

Organisational structure

- Top management; this includes the organisation's general manager and its executives
- Department managers; this includes the managers of software development, maintenance and software testing departments.
- Project managers; this is not only the project managers, but also team leaders of development projects and maintenance services.

Organisational framework

- Managers
  - Top managements executives
  - Software development and maintenance department managers
  - Software testing department managers
  - Project managers and team leaders of development and maintenance projects
  - Leaders of software testing teams

- Testers
  - Members of software testing team
- SQA professionals and interested practitioners
  - SQA trustees
  - SQA committee members
  - SQA forum members
  - SQA unit team members

Top Management

The Top managements overall responsibilities are as follows:

- Assuring the quality of company's software products and software maintenance services
- Communicating to employees at all levels the importance of product and service quality additionally to customer satisfaction
- Assuring satisfactory functioning and full conformity to customer requirements
- Ensuring that the SQA 's system objectives are established and realized
- Planning and overseeing implementation of changes for the SQA system adaptation to internal or external transformations (e.g. changes in clientele, competition or technology) (clientele: group of clients)
- Intervening directly to aid resolving crisis situations and minimize damages
- Ensuring availability of resources demanded by the SQA systems

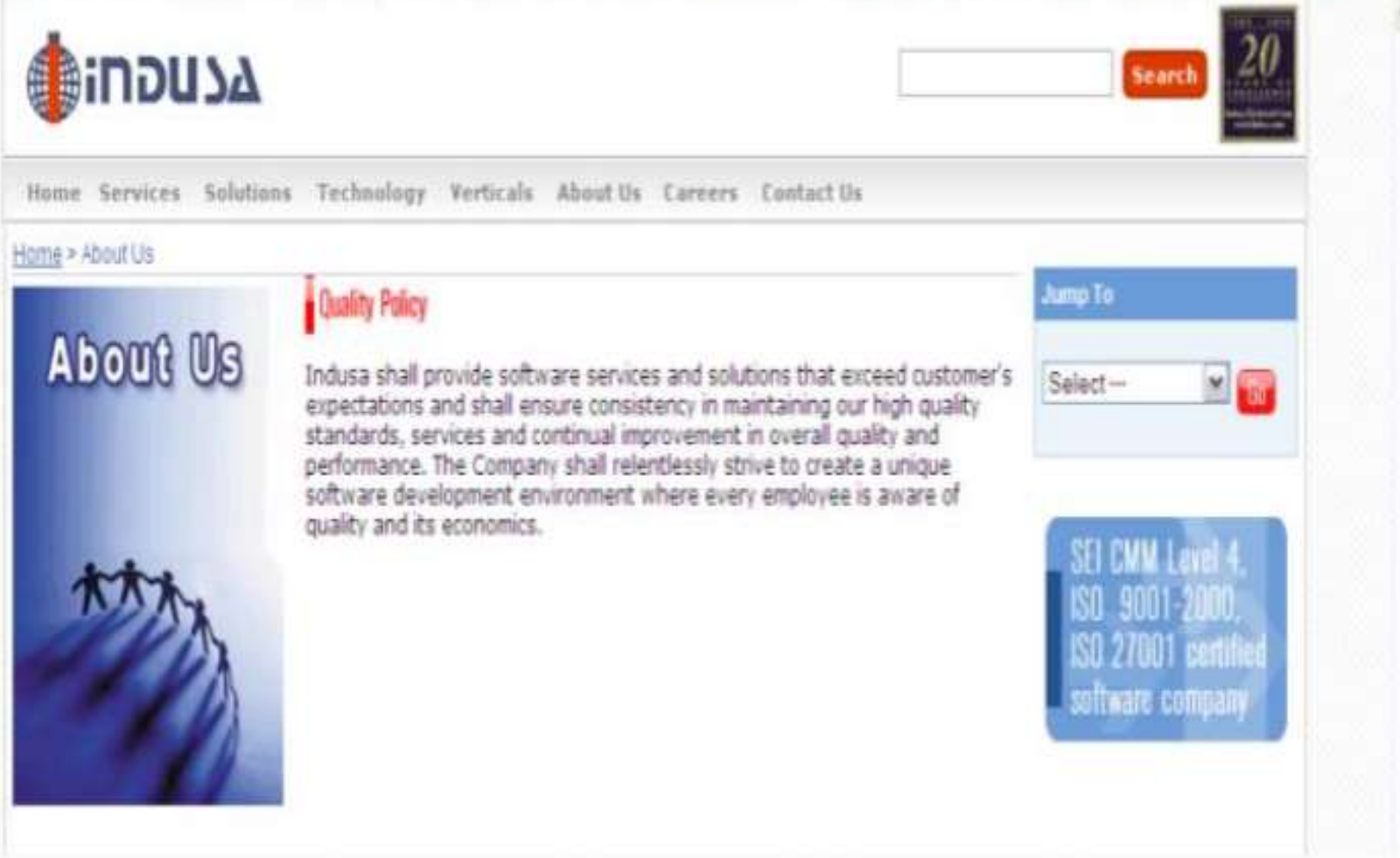
The Three Tools

- Establishment and updating of the organisation's software quality policy
- Assigning one of the executives in charge of software control issues
- Regular management reviews of performance with respect to software quality issues

Software quality policy

- Compliance with the originations goals and purpose
- Commitment to general software quality assurance concepts
- Commitment to quality standards
- Commitment to allocate adequate recourses for software quality assurance
- Commitment to continuous improvement of organisations quality and productivity.

Example of software quality policy



The executive in charge of software quality

- Takes responsibility of an annual activities program and budget
- Takes responsibility for the preparation of SQA system development plans
- Has overall control of annual SQA activities
- Present SQA issues to executive managers

Management review report

- Progress reports
- Periodic performance reports
- Customer satisfaction feedback
- Follow up reports
- Review of significant findings (finding is real/ reliable)

These are just some of the discussion points that come up in the management review

Department management

Quality system related responsibilities

- Preparation of the department's annual SQA activities and program budget
- Preparation of departments SQA system development plans
- Control of performance of the departments annual SQA program and development projects

Department management (2)

Some of the Project related responsibilities

- Control of compliance to quality assurance procedures in departments units
- Detailed follow up of contract review results and proposal approval
- Follow up of software tests and results
- Follow up of quality of maintenance services provision
- Follow up of the project risks and their solutions

Project management

Professional hands on tasks:

- Preparation of project and quality plans and their updates
- involvement in joint customer-supplier committee
- Close follow up of project team staffing

(staff hiring)

Project management (2)

Management tasks:

- deal with performance Review activities and consequent correction
- Address the performance of acceptance tasks (Assigned task is accepted and will be completed)
- undertake Software installation in customer sites and to show customer how to use software.
- Deal with SQA training and instruction of project team members
- Address customer requests and satisfaction of Customer

- Thank you