Chapter 8

Chapter 8: Measuring external product attributes: quality

Definition: Software Quality

ISO 8402 definition of quality:

- The totality of features and characteristics of a product or a service that bear on its ability to satisfy stated or implied needs
- Software Quality is defined to be a combination of features and functions of a product or service that determines the capability of software in order to satisfy the needs and requirements of software.
- Conformance to customers requirements
  - Containing fewer bugs
  - Providing better user satisfaction

What is Software Quality?

- Conventional definitions:
- Conformance to requirement
  - The requirements are clearly stated and the product must conform to it.
  - Any deviation from the requirements is regarded as a defect.
  - A good quality product contains fewer bugs.
- Fitness for use
  - Fit to user expectations: meet users needs.
  - A good quality product provides better user satisfaction.

What is Software Quality?

- **Various viewpoints/perspectives:**
- Conformance to customers requirements
- Requirement, design and development quality
- Process quality vs. end-product quality
  - Process quality: higher usability and dependability
  - End-product quality: less failure
- Internal vs. external characteristics
  - Relativity: advantage over similar products
  - Conformance: conformance to standards

Software Quality: Classification

- By observation:
  - Internal quality (quality while the product is being produced, including process and checks)
  - External quality (final product quality)
- By process:
  - Design quality
  - Implementation quality
  - Test quality
  - Maintenance quality

Software quality models: Boehm's model, McCall's model, ISO 9126 model, etc.

- Modeling Software Quality
- Selecting some attributes (or factors)
- Plotting relationship among attributes (many-to-
- many relationships) or
- Selecting some criteria (or intermediate and primitive constructs) to represent the attributes
- Mapping criteria (or primitive constructs) to metrics

Example: Boehm's Model

- Boehm software quality model was introduced in the year of 1978. The model is used to represent a hierarchical model that structures around high level characteristics, intermediate level characteristics, and primitive characteristics. All of these together results in to establishment of a high quality software model.
- The model defines the quality of software on the basis of a set of credentials and measurements. It is also elucidates a model of software quality characteristics. The high level of characteristics is made in such a way that answers following questions:

Cont…d

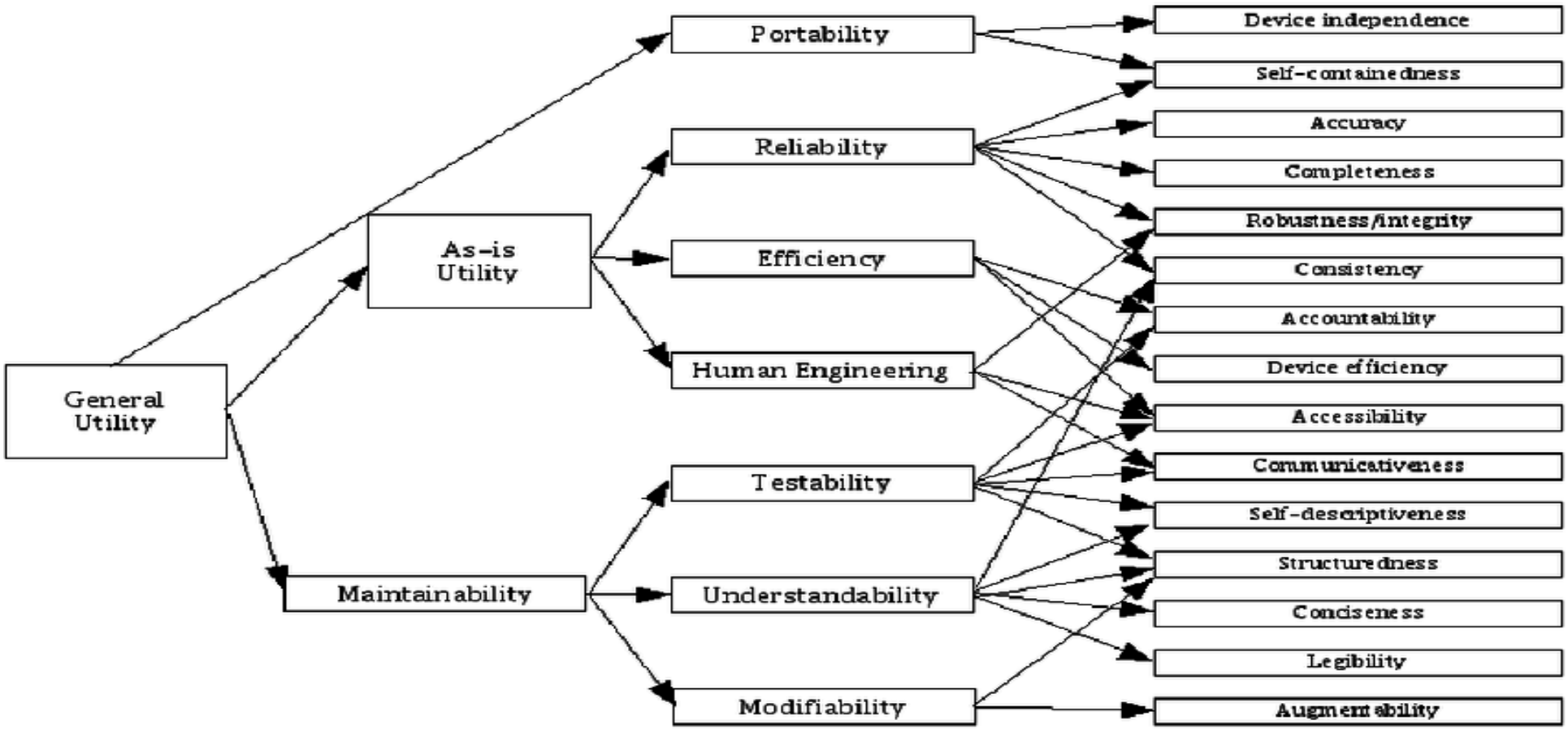
- **As-Is Utility**: It defines the way a utility signifies the as-is utility. It creates a question of how easily, reliably and efficiently an as can be utilized.
- **Maintainability**: This aspect decides how convenient it is to understand, change or re-evaluate a process.
- **Portability**: This aspect helps in deciding an effective way to change an environment.

The Boehm Model was created after doing improvements in McCall software quality model encountered few errors in it. This new model of Boehm was found to be more interesting as it is placed in hierarchical order. The order begins with addressing the major concerns of the end-users. On the contrary, the bottom of the hierarchy displays the technically inclined personnel.

Boehm model

- The model focuses on measuring properties and characteristics in such a way that create complex and non-technical stakeholders that are involved in the life-cycle of software. As compared to McCall model, this model is used in a widespread manner because of its bottom to top approach of software quality

Cont….. d


General Utility
As-is Utility
Maintainability
Portability
Reliability
Efficiency
Human Engineering
Testability
Understandability
Modifiability
Device independence
Self-containedness
Accuracy
Completeness
Robustness/integrity
Consistency
Accountability
Device efficiency
Accessibility
Communicativeness
Self-descriptiveness
Structuredness
Conciseness
Legibility
Augmentability

McCall's model

- This 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 factors.
- Product operation factors − Correctness, Reliability, Efficiency, Integrity, Usability.
- Product revision factors − Maintainability, Flexibility, Testability.
- Product transition factors − Portability, Reusability, Interoperability.

Product Operation Software Quality Factors

- According to McCall's model, product operation category includes five software quality factors, which deal with the requirements that directly affect the daily operation of the software. They are as follows −
- **Correctness**

These requirements deal with the correctness of the output of the software system. They include −

- Output mission
- The required accuracy of output that can be negatively affected by inaccurate data or inaccurate calculations.
- The completeness of the output information, which can be affected by incomplete data.
- The up-to- datedness of the information defined as the time between the event and the response by the software system.
- The availability of the information.
- The standards for coding and documenting the software system.

Product Operation Software Quality Factors (cont.…d)

- **Reliability**

Reliability requirements deal with service failure. They determine the maximum allowed failure rate of the software system, and can refer to the entire system or to one or more of its separate functions.

- **Efficiency**

It deals with the hardware resources needed to perform the different functions of the software system. It includes processing capabilities (given in MHz), its storage capacity (given in MB or GB) and the data communication capability (given in MBPS or GBPS).

It also deals with the time between recharging of the system's portable units, such as, information system units located in portable computers, or meteorological units placed outdoors.

- **Integrity**

This factor deals with the software system security, that is, to prevent access to unauthorized persons, also to distinguish between the group of people to be given read as well as write permit.

- **Usability**

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

Product Revision Quality Factors

According to McCall's model, three software quality factors are included in the product revision category. These factors are as follows −

- Maintainability

This factor considers 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.

- Flexibility

This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software. These include adapting the current software to additional circumstances and customers without changing the software. This factor's requirements also support perfective maintenance activities, such as changes and additions to the software in order to improve its service and to adapt it to changes in the firm's technical or commercial environment.

- Testability

Testability requirements deal with the testing of the software system as well as with its operation. It includes predefined intermediate results, log files, and also the automatic diagnostics performed by the software system prior to starting the system, to find out whether all components of the system are in working order and to obtain a report about the detected faults. Another type of these requirements deals with automatic diagnostic checks applied by the maintenance technicians to detect the causes of software failures.

Product Transition Software Quality Factor

According to McCall's model, three software quality factors are included in the product transition category that deals with the adaptation of software to other environments and its interaction with other software systems. These factors are as follows −

- Portability

Portability requirements tend to the adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth. The software should be possible to continue using the same basic software in diverse situations.

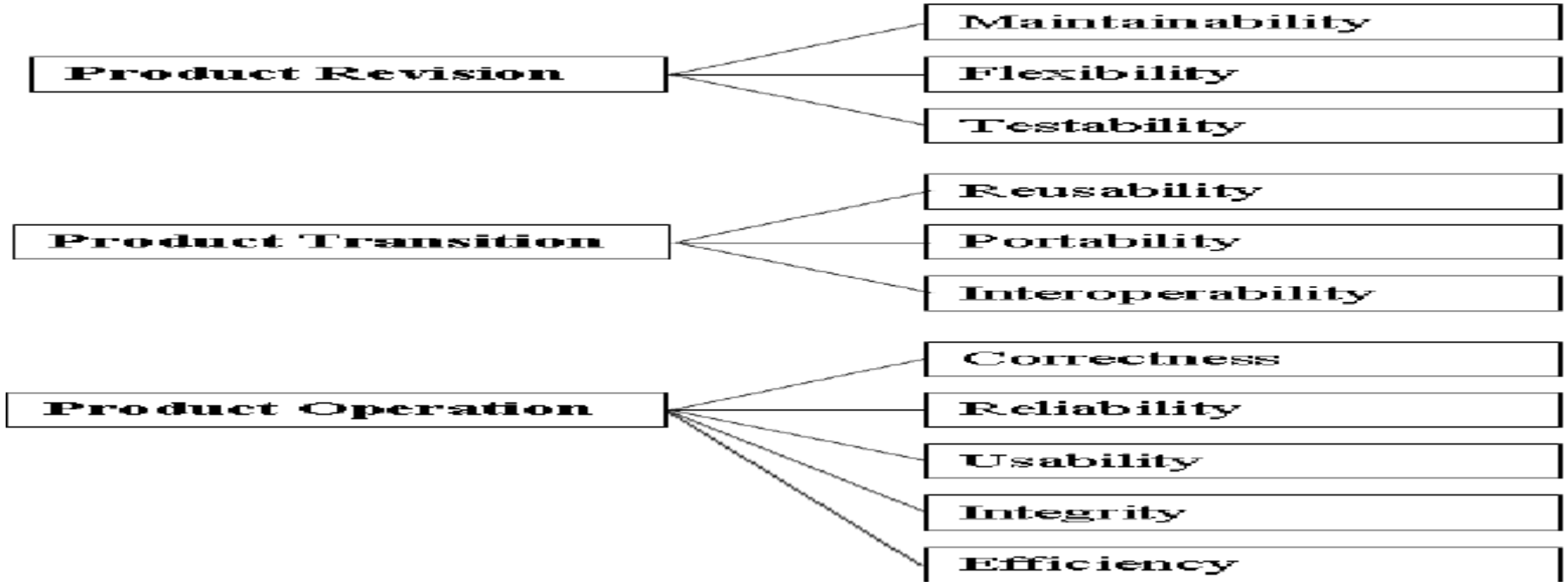
- Reusability

This factor deals with the use of software modules originally designed for one project in a new software project currently being developed. They may also enable future projects to make use of a given module or a group of modules of the currently developed software. The reuse of software is expected to save development resources, shorten the development period, and provide higher quality modules.

- 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.

McCall's model software quality



Software quality metrics

- Software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.
- Software quality metrics can be further divided into three categories
  - Product quality metrics
  - In-process quality metrics
  - Maintenance quality metrics

Product Quality Metrics

- This metrics include the following −
  - Mean Time to Failure
  - Defect Density
  - Customer Problems
  - Customer Satisfaction

Cont…d

- Mean Time to Failure

It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

- Defect Density

It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.

- Customer Problems

It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

- The problems metric is usually expressed in terms of Problems per User-Month (PUM).

Cont...d

- Customer Satisfaction

Customer satisfaction is often measured by customer survey data through the five-point scale −

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys. Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis. For example −

- Percent of completely satisfied customers
- Percent of satisfied customers
- Percent of dis-satisfied customers
- Percent of non-satisfied customers

Usually, this percent satisfaction is used.

In-process Quality Metrics

In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes −

- Defect density during machine testing
- Defect arrival pattern during machine testing
- Phase-based defect removal pattern
- Defect removal effectiveness

Defect density during machine testing

- Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.
- This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.

Cont…d

Defect arrival pattern during machine testing

The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following −

- The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.
- The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.
- The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

Cont…d

Phase-based defect removal pattern

This is an extension of the defect density metric during testing. In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.

Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software. The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

Cont…d

Defect removal effectiveness

It can be defined as follows −

$$DRE = \frac{Defect\ removed\ during\ a\ development\ phase}{Defects\ latent\ in\ the\ product} \times 100\%$$

This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called early defect removal when used for the front-end and phase effectiveness for specific phases. The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

Maintenance Quality Metrics

Although much cannot be done to alter the quality of the product during this phase, following are the fixes that can be carried out to eliminate the defects as soon as possible with excellent fix quality.

- Fix backlog and backlog management index
- Fix response time and fix responsiveness
- Percent delinquent fixes
- Fix quality

Cont...d

Fix backlog and backlog management index

Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.

Backlog Management Index (BMI) is used to manage the backlog of open and unresolved problems.

$$DRE = \frac{Defect\ removed\ during\ a\ development\ phase}{Defects\ latent\ in\ the\ product} \times 100\%$$

If BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

Fix response time and fix responsiveness

The fix response time metric is usually calculated as the mean time of all problems from open to close. Short fix response time leads to customer satisfaction.

The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.

Percent delinquent fixes

- It is calculated as follows −

$$Percent\ Delinquent\ Fixes = \frac{Number\ of\ fixes\ that\ exceeded\ the\ response\ time\ criteria\ by\ ceverity\ level}{Number\ of\ fixes\ delivered\ in\ a\ specified\ time} \times 100\%$$

Fix Quality

- Fix quality or the number of defective fixes is another important quality metric for the maintenance phase. A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction. The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.
- A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered. The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.
- Usually the longer the latency, the more will be the customers that get affected. If the number of defects is large, then the small value of the percentage metric will show an optimistic picture. The quality goal for the maintenance process, of course, is zero defective fixes without delinquency

Measuring customer satisfaction

These terms are often mistakenly equated. In fact, when a client doesn’t feel good about the results of cooperation, it doesn’t necessarily mean a delivered product is of low quality. There are so many factors that are able to influence customers’ opinion on partnering with a software development company.

For instance, IT engineers managed to create an outstanding application but failed to deliver it in time, and communication with a client wasn’t smooth enough. These issues can result in an unfavorable impression regardless of an excellent software solution. Thus, a high-quality product isn’t the only factor to ensure customers’ positive testimonials and references.

The best way to keep customers satisfied is to complete work within the time frame, meet the budget, and develop a great product at the end. In words it seems a piece of cake but what about in practice? What exactly you should do to win clients’ loyalty? To find the answers, you have to dive into the peculiarities of measuring customer satisfaction.

Cont…d

- 1. Conduct regular surveys and ensure close cooperation

Make this a habitual daily part of your project plan. Ask more questions during meetings and video-conferences to gather sufficient information about the client’s expectations. This simple method allows detecting a problem before its actual occurrence. You can also challenge your company’s quality manager to draw up a list of questions to quickly understand whether a client is happy with a service or not.

- 2. Quality assurance

Bring your quality management system in line with ISO 9001:2000 standard. It allows monitoring of the products quality, services, and other processes in an IT company. Credible information about the real quality of the company’s services and products helps flexibly manage production processes, monitor and resolve problem cases, and avoid the same mistakes in future. Day-to-day quality monitoring helps objectively evaluate the level of services.

Cont…d

3. Monitoring user feedback

It’s important to constantly monitor the fate of a solution, even if a client is a product owner and takes care of further project development and promotion. It’s the task for a marketing department to gather end-user reviews and analyze them. Collecting this information allows understanding users’ attitude towards a software application and a company which created this solution.

4. Customer survey after completing a project

- To get the full picture of customers’ attitude to service quality, it’s better to use not only one questionnaire but a complex of them. Customer Satisfaction Score (CSAT) is one of the most popular methods of measuring customer satisfaction. Ask clients to rate the quality of the services they received from your company, e.g. by giving points from 1 to 10.
- Customer Effort Score (CES) is a survey allowing you to find out client’s overall impression about working with you. Ask “how easy was it to solve your problem with our assistance?” and provide possible answers.
- Net Promoter Score (NPS) helps to understand how loyal a customer is. Create a survey with a question similar to “how likely is that you will recommend our company to somebody?”. This is how you will define the clients who will probably come back to you with new projects.

Software Quality Assurance (SQA)

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.

SQA practices are implemented in most types of software development, regardless of the underlying software development model being used. SQA incorporates and implements software testing methodologies to test the software. Rather than checking for quality after completion, SQA processes test for quality in each phase of development, until the software is complete. With SQA, the software development process moves into the next phase only once the current/previous phase complies with the required quality standards. SQA generally works on one or more industry standards that help in building software quality guidelines and implementation strategies.

It includes the following activities −

- Process definition and implementation
- Auditing
- Training

Cont…d

Processes could be −

- Software Development Methodology
- Project Management
- Configuration Management
- Requirements Development/Management
- Estimation
- Software Design
- Testing, etc.

Once the processes have been defined and implemented, Quality Assurance has the following responsibilities −

- Identify the weaknesses in the processes
- Correct those weaknesses to continually improve the process