ELECTIVE III: SOFTWARE QUALITY ASSURANCE

SUB. CODE: 18MIT41E

Unit V:

Software quality metrics: classification of software quality metrics – process metrics – product metrics – implementation of software quality metrics – quality management standards: ISO 9001 and ISO 9000-3 – certification according to ISO 9000- 3 – Capability Maturity Model and CMMI assessment methodology. SQA project process standards – IEEE software engineering standards: IEEE Std 1012 – verification and validation – IEEE Std 1028 – reviews.

Software Quality Metrics

Software metrics can be classified into three categories −

- **Product metrics** − Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
- **Process metrics** − These characteristics can be used to improve the development and maintenance activities of the software.
- **Project metrics** − This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

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

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)**.

```
PUM = Total Problems that customers reported (true defect and non-defect oriented problems) for a time period + Total number of license months of the software during the period
```

Where,

```
Number of license-month of the software = Number of install license of the software × Number of months in the calculation period
```

PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

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.

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.

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.

Defect removal effectiveness

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

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.

BMI=NumberofproblemsclosedduringthemonthNumberofproblemsarrivedduringthemonth×100%BMI=NumberofproblemsclosedduringthemonthNumberofproblemsarriveddduringthemonth×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.

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.

Measurement is of two types −

- Direct measurement
- Indirect measurement

Direct Measurement

These are the measurements that can be measured without the involvement of any other entity or attribute.

The following direct measures are commonly used in software engineering.

- Length of source code by LOC
- Duration of testing purpose by elapsed time
- Number of defects discovered during the testing process by counting defects
- The time a programmer spends on a program

Indirect Measurement

These are measurements that can be measured in terms of any other entity or attribute.

Measurement for Prediction

For allocating the appropriate resources to the project, we need to predict the effort, time, and cost for developing the project. The measurement for prediction always requires a mathematical model that relates the attributes to be predicted to some other attribute that we can measure now. Hence, a prediction system consists of a mathematical model together with a set of prediction procedures for determining the unknown parameters and interpreting the results.

Measurement scales are the mappings used for representing the empirical relation system. It is mainly of 5 types −

- Nominal Scale
- Ordinal Scale

- Interval Scale
- Ratio Scale
- Absolute Scale

Software Measurement Metrics

Software metrics is a standard of measure that contains many activities which involve some degree of measurement. It can be classified into three categories: product metrics, process metrics, and project metrics.

- **Product metrics** describe the characteristics of the product such as size, complexity, design features, performance, and quality level.
- **Process metrics** can be used to improve software development and maintenance. Examples include the effectiveness of defect removal during development, the pattern of testing defect arrival, and the response time of the fix process.
- **Project metrics** describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

Scope of Software Metrics

Software metrics contains many activities which include the following −

- Cost and effort estimation
- Productivity measures and model
- Data collection
- Quantity models and measures
- Reliability models
- Performance and evaluation models
- Structural and complexity metrics
- Capability – maturity assessment
- Management by metrics
- Evaluation of methods and tools

Software measurement is a diverse collection of these activities that range from models predicting software project costs at a specific stage to measures of program structure.

Software metrics is a standard of measure that contains many activities, which involves some degree of measurement. The success in the software measurement lies in the quality of the data collected and analyzed.

What is Good Data?

The data collected can be considered as a good data, if it can produce the answers for the following questions −

- **Are they correct? −** A data can be considered correct, if it was collected according to the exact rules of the definition of the metric.
- **Are they accurate? −** Accuracy refers to the difference between the data and the actual value.
- **Are they appropriately precise? −** Precision deals with the number of decimal places needed to express the data.
- **Are they consistent? −** Data can be considered as consistent, if it doesn't show a major difference from one measuring device to another.
- **Are they associated with a particular activity or time period? −** If the data is associated with a particular activity or time period, then it should be clearly specified in the data.
- **Can they be replicated? −** Normally, the investigations such as surveys, case studies, and experiments are frequently repeated under different circumstances. Hence, the data should also be possible to replicate easily.

How to Define the Data?

Data that is collected for measurement purpose is of two types −

- **Raw data −** Raw data results from the initial measurement of process, products, or resources. For example: Weekly timesheet of the employees in an organization.
- **Refined data −** Refined data results from extracting essential data elements from the raw data for deriving values for attributes.

Data can be defined according to the following points −

- Location
- Timing
- Symptoms
- End result
- Mechanism
- Cause
- Severity
- Cost

How to Collect Data?

Collection of data requires human observation and reporting. Managers, system analysts, programmers, testers, and users must record row data on forms. To collect accurate and complete data, it is important to −

- Keep procedures simple
- Avoid unnecessary recording
- Train employees in the need to record data and in the procedures to be used

- Provide the results of data capture and analysis to the original providers promptly and in a useful form that will assist them in their work
- Validate all data collected at a central collection point

Planning of data collection involves several steps −

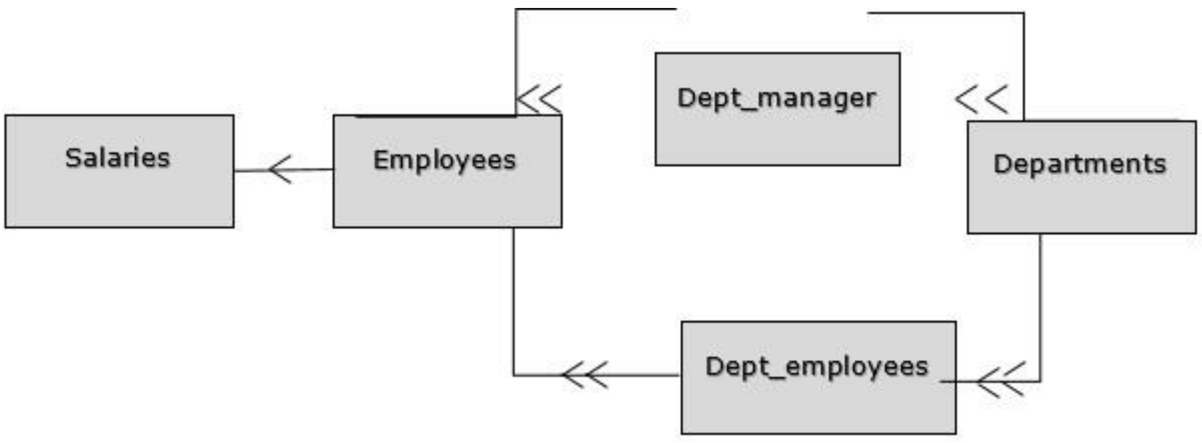
- Decide which products to measure based on the GQM analysis
- Make sure that the product is under configuration control
- Decide exactly which attributes to measure and how indirect measures will be derived
- Once the set of metrics is clear and the set of components to be measured has been identified, devise a scheme for identifying each activity involved in the measurement process
- Establish a procedure for handling the forms, analyzing the data, and reporting the results

Data collection planning must begin when project planning begins. Actual data collection takes place during many phases of development.

For example − Some data related to project personnel can be collected at the start of the project, while other data collection such as effort begins at project starting and continues through operation and maintenance.

How to Store and Extract Data

In software engineering, data should be stored in a database and set up using a Database Management System (DBMS). An example of a database structure is shown in the following figure. This database will store the details of different employees working in different departments of an organization.



In the above diagram, each box is a table in the database, and the arrow denotes the many-to-one mapping from one table to another. The mappings define the constraints that preserve the logical consistency of the data.

Once the database is designed and populated with data, we can make use of the data manipulation languages to extract the data for analysis.

Standards and Certificates

Several national and international standards institutes, professional and industry-oriented organizations have been involved in the development of SQA standards.

The following institutes and organizations are the main developers of SQA and software engineering standards −

- IEEE (Institute of Electrical and Electronics Engineers) Computer Society
- ISO (International Organization for Standardization)
- DOD (US Department of Defense)
- ANSI (American National Standards Institute)
- IEC (International Electro Technical Commission)
- EIA (Electronic Industries Association)

These organizations provide updated international standards to the quality of professional and managerial activities performed in software development and maintenance organizations.

They also provide SQA certification through independent professional quality audits. These external audits assess achievements in the development of SQA systems and their implementation. Certification, which is granted after the periodic audits, will be valid only until the next audit, and therefore must be renewed. At present, the ISO 9000 Certification Service is the most prominent provider of SQA certification in Europe and other countries.

They also provide the tools for self-assessment of an organization's SQA system and its operation. The Capacity Maturity Model (CMM) developed by the Software Engineering Institute (SEI), Carnegie Mellon University, and ISO/IEC Std 15504 are the examples of this approach.

SQA Standards

Software quality assurance standards can be classified into two main classes −

- Software quality assurance management standards, including certification and assessment methodologies (quality management standards)
- Software project development process standards (project process standards)

Quality Management Standards

These focus on the organization's SQA system, infrastructure and requirements, while leaving the choice of methods and tools to the organization. With quality management standards, organizations can steadily assure that their software products achieve an acceptable level of quality.

Example − ISO 9000-3 and the Capability Maturity Model (CMM)

Project Process Standards

These focus on the methodologies for implementing the software development and maintenance projects. These standards include the following –

- The steps to be taken
- Design documentation requirements
- Contents of design documents
- Design reviews and review issues
- Software testing to be performed
- Testing topics

Naturally, due to their characteristics, many SQA standards in this class can serve as software engineering standards and vice versa.

The characteristics of these two classes of standards are summarized in the following table.

Characteristics	**Quality Management Standards**	**Project Process Standards**
The target unit	Management of software development, maintenance and the specific SQA units	A software development and maintenance project team
The main focus	Organization of SQA systems, infrastructure and requirements	Methodologies for carrying out software development and maintenance projects
The standard's objective	"What" to achieve	"How" to perform
The standard's goal	Assuring the supplier's software quality and assessing its software process capability	Assuring the supplier's software quality and assessing its software process capability Assuring the quality of a specific software project.
Examples	ISO 9000-3 SEI's CMM	ISO/IEC 12207 IEEEStd 1012-1998

ISO 9001 Certification

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies. ISO technical committees prepare the International Standards. ISO collaborates closely with the International Electro-technical Commission (IEC) on all matters of electro-technical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2. Draft of the International Standards adopted by the technical committees is circulated to the member bodies for voting. ISO 9001 was prepared by

Technical Committee ISO/TC 176, Quality management and quality assurance, Subcommittee SC 2, Quality systems.

Process Approach

This International Standard promotes the adoption of a process approach when developing, implementing, and improving the effectiveness of a quality management system, to enhance customer satisfaction by meeting the customer requirements. For an organization to function effectively, it has to determine and manage numerous linked activities. An activity or set of activities using resources, and managed in order to enable the transformation of inputs into outputs, can be considered as a process.

Often the output from one process directly forms the input to the next. The application of a system of processes within an organization, together with the identification and interactions of these processes, and their management to produce the desired outcome, can be referred to as the **"process approach"**.

An advantage of the process approach is the ongoing control that it provides over the linkage between the individual processes within the system of processes, as well as over their combination and interaction. When used within a quality management system, such an approach emphasizes the importance of the following −

- Understanding and meeting the requirements
- Need to consider the processes in terms of added value
- Obtain the results of process performance and effectiveness
- Continual improvement of processes based on objective measurement

ISO 9001 - Application to Software: the TickIT Initiative

TickIT was launched in the late 1980s by the UK software industry in cooperation with the UK Department for Trade and Industry to promote the development of a methodology for adapting ISO 9001 to the characteristics of the software industry known as the TickIT initiative.

TickIT is, additionally, specializing in information technology (IT). It covers the entire range of commercial software development and maintenance services. TickIT, now managed and maintained by the DISC Department of BSI (the British Standards Institute), is accredited for the certification of IT organizations in the UK and Sweden.

Its activities include −

- Publication of the TickIT Guide, which supports the software industry's efforts to spread ISO 9001 certification. The current guide (edition 5.0, TickIT, 2001), which includes references to ISO/IEC 12207 and ISO/IEC 15504, is distributed to all TickIT customers.
- Performance of audit-based assessments of software quality systems and consultation to organizations on the improvement of software development and maintenance processes in addition to their management.
- Conduct ISO 9000 certification audits.

TickIT auditors who conduct audit-based assessments and certification audits are registered by the International Register of Certificated Auditors (IRCA). Registered IRCA auditors are required, among other things, to have experience in management and software development; they must also successfully complete an auditor's course.

Registered lead auditors are required to have a demonstrated experience in conducting and directing TickIT audits.

Software Process Assessment

A software process assessment is a disciplined examination of the software processes used by an organization, based on a process model. The assessment includes the identification and characterization of current practices, identifying areas of strengths and weaknesses, and the ability of current practices to control or avoid significant causes of poor (software) quality, cost, and schedule.

A software assessment (or audit) can be of three types.

- A **self-assessment (first-party assessment)** is performed internally by an organization's own personnel.
- A **second-party assessment** is performed by an external assessment team or the organization is assessed by a customer.
- A **third-party assessment** is performed by an external party or (e.g., a supplier being assessed by a third party to verify its ability to enter contracts with a customer).

Software process assessments are performed in an open and collaborative environment. They are for the use of the organization to improve its software processes, and the results are confidential to the organization. The organization being assessed must have members on the assessment team.

Software Process Maturity Assessment

The scope of a software process assessment can cover all the processes in the organization, a selected subset of the software processes, or a specific project. Most of the standard-based process assessment approaches are invariably based on the concept of process maturity.

When the assessment target is the organization, the results of a process assessment may differ, even on successive applications of the same method. There are two reasons for the different results. They are,

- The organization being investigated must be determined. For a large company, several definitions of organization are possible and therefore the actual scope of appraisal may differ in successive assessments.
- Even in what appears to be the same organization, the sample of projects selected to represent the organization may affect the scope and outcome.

When the target unit of assessment is at the project level, the assessment should include all meaningful factors that contribute to the success or failure of the project. It should not be limited by established dimensions of a given process maturity model. Here the degree of implementation and their effectiveness as substantiated by project data are assessed.

Process maturity becomes relevant when an organization intends to embark on an overall long-term improvement strategy. Software project assessments should be independent assessments in order to be objective.

Software Process Assessment Cycle

According to Paulk and colleagues (1995), the CMM-based assessment approach uses a six-step cycle. They are −

- Select a team - The members of the team should be professionals knowledgeable in software engineering and management.
- The representatives of the site to be appraised complete the standard process maturity questionnaire.
- The assessment team performs an analysis of the questionnaire responses and identifies the areas that warrant further exploration according to the CMM key process areas.
- The assessment team conducts a site visit to gain an understanding of the software process followed by the site.
- The assessment team produces a list of findings that identifies the strengths and weakness of the organization's software process.
- The assessment team prepares a Key Process Area (KPA) profile analysis and presents the results to the appropriate audience.

For example, the assessment team must be led by an authorized SEI Lead Assessor. The team must consist of between four to ten team members. At least, one team member must be from the organization being assessed, and all team members must complete the SEI's Introduction to the CMM course (or its equivalent) and the SEI's CBA IPI team training course. Team members must also meet some selection guidelines.

With regard to data collection, the CBA IPI relies on four methods −

- The standard maturity questionnaire
- Individual and group interviews
- Document reviews
- Feedback from the review of the draft findings with the assessment participants

SCAMPI

The Standard CMMI Assessment Method for Process Improvement (SCAMPI) was developed to satisfy the CMMI model requirements (Software Engineering Institute, 2000). It is also based on the CBA IPI. Both the CBA IPI and the SCAMPI consist of three phases −

- Plan and preparation
- Conduct the assessment onsite
- Report results

The activities for the plan and preparation phase include −

- Identify the assessment scope
- Develop the assessment plan
- Prepare and train the assessment team

- Make a brief assessment of participants
- Administer the CMMI Appraisal Questionnaire
- Examine the questionnaire responses
- Conduct an initial document review

The activities for the onsite assessment phase include −

- Conduct an opening meeting
- Conduct interviews
- Consolidate information
- Prepare the presentation of draft findings
- Present the draft findings
- Consolidate, rate, and prepare the final findings

The activities of the reporting results phase include −

- Present the final findings
- Conduct an executive session
- Wrap up the assessment

Testing Objectives and Principles

The testing objective is to test the code, whereby there is a high probability of discovering all errors. This objective also demonstrates that the software functions are working according to software requirement specifications (SRS) with regard to functionality, features, facilities and performance. It should be another certification is given by ISO, which also specifies the quality management infrastructure required to achieve the best quality performance.

ISO 9000 Quality System

The ISO 9000 standards are a collection of formal International Standards, technical reports, handbooks and web based documents on quality management and quality assurance. ISO technical committee and web based documents on quality management and quality assurance. ISO technical committee and its sub committees are responsible for the development of the standards.

The work is conducted on the basis of “consensus” among quality and industry experts nominated by the national standards bodies, representing a wide range of interested parties. The ISO 9000 series of standards are generic rather than industry specific. It can be applied to any organization, large or small, whether its product, and whether it is a business enterprise, a public administration, or a government department. ISO 9000 is a family of standards for quality management systems.

ISO 9000 is maintained by ISO, the International Organization for Standardization and is administered by accreditation and certification bodies. The rules are updated, the time and changes in the requirements for quality, motive change. Recently, on November 15, 2008, has made changes to the requirements of ISO 9001.

Some of the requirements in 9001 9 which is one of the standards in the ISO 9000 family) include

• A set of procedures that cover all key processes in the business; • Monitoring processes to ensure they are effective;

• Keeping adequate records;

• Checking output for defects, with appropriate and corrective action where necessary;

• Regularly reviewing individual processes and the quality system itself for effectiveness; and

• Facilitating continual improvement A company or organization that has been independently audited and certified to be in conference with ISO 9001 may publicly state that it is "ISO 9001 registered."

Certification to an ISO 9001 standard does not guarantee any quality of end products and services; rather, it certifies that formalized business processes are being applied. Although the standards originated in manufacturing, they are now employed across several types of organizations. A "product", in ISO vocabulary, can mean a physical object, services, or software.

ISO 9000 Standards

• **ISO 9001: 2008 Quality management systems —** Requirements is intended for use in any organization regardless of size, type or product (including service). It provides a number of requirements which an organization needs to fulfill to achieve customer satisfaction through consistent products and services which meet customer expectations. It includes a requirement for continual (i.e., planned) improvement of the Quality Management System, for which ISO 9004:2004 provides many hints. This is the only implementation for which third party auditors can grant certification. It should be noted that certification is not described as any of the 'needs' of an organization as a driver for using ISO 9001 but does recognize that it may be used for such a purpose.

• **ISO 9004:2000 Quality management systems -** Guidelines for performance improvements covers continual improvement. This gives you advice on what you could do to enhance a mature system. This document very specifically states that it is not intended as a guide to implementation. There are many more standards in the ISO 9001 series, many of them not even carrying "ISO 9000" numbers.

For example, some standards in the 10,000 range are considered part of the 9000 group: ISO 10007: 1995 discusses configuration management, which for most organizations is just one element of a complete management system. The emphasis on certification tends to overshadow the fact that there is an entire family of ISO 9000 standards… Organizations stand to obtain the greatest value when the standards in the new core series are used in an integrated manner, both with each other and with the other standards making up the ISO 9000 family as a whole. Note that the previous members of the ISO 9000 series 9002 and 9003 have been integrated into 9001. In most cases, an organization claiming to be "ISO 9000 registered" is referring to ISO 9001.

ISO 9001 Scope of Software Quality

The ISO 9001 standard has 20 clauses 4.1 to 4.20 that lay down guidelines for the development of quality assurance systems. These guidelines define the essential features of the software quality management system and suggest controls and methods that allow the software to meet customer needs.

The steps that have to be taken to set up the ISO 9001 standard are :

1. A commitment to quality by the highest level of management of the organization and necessary resource allocation. Understanding quality requirements for organizing the work.
2. Identifying procedures to develop the software and to test whether it can meet customer needs in a defined time.
3. Acceptance of software by customer.
4. Delivery and installation.
5. Maintenance.
6. Support activities, namely, documentation control, maintenance of detailed records , and training.
7. Non-conformity control and corrective action these steps when implemented through a set of procedures will make the company eligible for ISO 9001.

ISO 9001 gives a detailed explanation for each clause, which requires compliance for a company to become eligible for ISO 9001 certification. If all the above steps are fully executed, then clauses 4.1 to 4.20 of 4.20 of ISO 9001 are taken care of. ISO 9001 does not recommend standard formats but demands execution of these steps.

ISO 9001 Checklist

Quality Policy (4.0 of ISO 9001) The management shall define the quality policy and ensure all concerned understand this policy. The policy document should be signed by the CEO and be displayed at prominent places.

Management Review (4.1 to ISO 9001) The quality system adopted to satisfy the requirements for this information standard shall be reviewed periodically to ensure continuity, suitability, and effectiveness. The company shall appoint a management representative with the responsibility to ensure that ISO guidelines are implemented.

Quality System Procedure (QSP) (4.2 of ISO 9001) The company shall establish a OSP and maintain a documented quality system as a means of ensuring that it confirms to the specified requirements of ISO.

(a) Preparation of documented quality system procedures and instructions.

(b) Effective implementation of these procedures.

Broadly, QAS includes the following:

- QA manual
- Management procedures
- Technical work instructions

Contract Review (4.3 of ISO 9001)

The company shall establish a procedure for contract review and coordination of activities:

- Contractual requirements are adequately defined and documented
- Procedure to resolve any deviation
- Ensure capability to meet contractual requirement
- Records of contract review are maintained

Design control (4.4 of/ so 9001) Establish and maintain procedures to control and verify the design to ensure that specified requirements are met.

The points to be covered are

- Identification of design consideration
- Design methodology
- Use of past design experience
- Inclusion of design experience
- Inclusion of design Inputs requirements
- Design verification
- Design changes

Document control (4.5 0f ISO 9001)

- Design documents
- Planning documents
- Procedural documents
- Reference documents
- Document master list and its revision log
- The organization for documentation

Purchasing (4.6 of ISO 9001)

- Hardware, software, tools
- Assessment of sub-contractors
- Verification of supplies

Software Identification and Tracing (4.7 of ISO 9001)

Establish a procedure whereby software can be drilled down to all documentation to trace any information needed.
Appropriate numbering, coding and version control numbers should be introduced.

Purchase supplied Product (4.7 of ISO 9001)
The company shall establish a procedure to receive, verify, store and maintain client supplied software/ tools etc. These items are part of the deliverables of the company. If software is found unsuitable it should be returned with resource for rejection.

Process control (4.9 of ISO 9001) A procedure to ensure that from development to implementation all activities are carried out to meet the quality requirements. The procedure should have following aspects covered through documents: • Work Instruction • Monitoring and control of activities • Approval of processes and hardware and software/ tools etc. •

Workmanship standards Inspection and testing (4.10 ISO 9001) Establish a procedure to ensure that resources are as per the required standards. It should cover • In process inspection and testing • Final inspection and testing • Records of testing **9001)** • Establish a procedure to ensure that rules, practices, convention
Inspection, measuring and testing equipment (4.11 of ISO s, tools and techniques agreed upon between the company and customer are adhered to and used in the development • Software tools, e.g., compilers, editors, database software, steps, communication software internet and Web tools are evaluated earlier for purchase and installation • Ensure that test plans are evolved for all stages of development

Test status (4.12 of ISO 9001) Set a procedure to ensure that only tested and quality assured products are dispatched, used or installed.

Control of Non- conforming Products (4.13 0f ISO 9001) Set up a procedure to ensure that the software not conforming to specified requirements is not dispatched, installed or used by mistake.

Corrective Action (4.14 of ISO 9001) Set up a procedure to ensure that corrective action is taken to set right non- conforming software.

Handling, Storage and delivery (4.15 ISO 9001) Ensure that unauthorized persons do not tamper with the software during development and after completion, and as a result, deliver software that has quality problems. Strict controls on access and us of media are necessary to control the quality of outgoing software.

Quality Records (4.16 of ISO 9001) A procedure should be set to identify and maintain quality records with a clear and unique relation to the software for which it is maintained. There should be a set of guidelines for retention of quality records for future reference in case a dispute arises. The retention period is of mutual convenience and as per contract terms, if any.

Internal quality Audits (4.17 of IO 9001) A procedure should be in place to audit whether the quality procedures have been complied with. The audit will be scheduled and results reported and corrective action taken as laid down.

Training (4.18 of ISO 9001) The company will maintain a procedure to identify the training needs of people responsible for quality assurance. The training records will justify the inclusion of personnel in the testing team.

Software Maintenance Service (4.19 0f ISO 9001) If software includes support and service needs of the customer, then a support procedure should be in place that ensures that this service is effectively carried out.

Statistical Techniques (4.20 of ISO 9001) A procedure should be in place be in place to verify the needs for statistical techniques that will in turn verify process capability and characteristics. This would be more applicable in process control software.

Advantages of ISO 9000

It is widely acknowledged that proper quality management improves business, often having a positive effect on investment, market share, sales growth, sales margins, competitive advantage, and avoidance of Litigation. The quality principles in ISO 9000: 2000 are also sound, according to Wade, and Barnes, who says "ISO 9000 guidelines provide a comprehensive model for quality management systems that can make any company competitive." Barnes also cites a survey by LIoyd's Register Quality Assurance which indicated that ISO 9000 increased net profit, and another by Deloitte Touche which reported that the costs of registration were recovered in three years.

According to the Providence Business News, implementing ISO often gives the following advantages:

1. Create a more efficient , effective operation
2. Increase customer satisfaction and retention
3. Reduce audits
4. Enhance marketing
5. Improve employee motivation, awareness, and morale
6. Promote international trade
7. Increase profit
8. Reduce waste and increases productivity

However, a broad statistical study of 800 Spanish companies found that ISO 9000 registration in itself creates little improvement because companies interested in it have usually already made some type of commitment to quality management and were performing just as well before registration. In today's service sector driven economy, more and more companies are using ISO 9000 as a business tool. Through the use of properly stated quality objectives, customer satisfaction surveys and a well-defined continual improvement program companies are using IO 9000 processes to increase their efficiency and profitability.

Problems of ISO 9001

A common criticism of ISO 9001 is the amount of money, time and paperwork required for registration. According to Barnes, “Opponents claim that it is only for documentation. Proponents believe that if a company has documented its quality systems, then most of the paperwork has already been completed.” According to Seddon, ISO promotes specification, control, and procedures rather than understanding and improvement. Ade argues that IO 9000 is effective as a guideline, but that is promoting it as a standard “helps to mislead companies into thinking that certification means better quality, [undermining] the need for an organization to set its own quality standards“. Reliance on the specifications of ISO 9001 does not guarantee a successful quality system. The standard is seen as especially prone to failure when a company is interested in certification before quality. Certifications are in fact often based on customer contractual requirements rather than a desire to actually improve quality. “ If you just want the certificate on the wall, chances are, you will create a paper system that doesn’t have much to do with the way you actually run your business,’” said ISO’s Roger Frost. Certification by an independent auditor is often seen as the problem area, and according to Barnes, “has become a vehicle to increase consulting services.” In fact, ISO itself, advises that ISO 9001 can be implemented without certification, simply for the quality benefits that can be achieved. Another problem reported is the competition among the numerous certifying bodies, leading to a softer approach to the defects noticed in the operation of the Quality system of a firm.

Capability Maturity Model Integration (CMMI)

CMMI is a collection of best practices that met the needs of organizations in different areas of interest. A collection of best practices that cover a particular area of interest is called a CMMI model. **Capability maturity model Integration (CMMI)** in software engineering and organizational development is a process improvement approach that provides organizations with the essential elements for effective process improvement.

It can be used to guide process improvement across a project, a division, or an entire organization. CMMI helps integrate traditionally separate organizational functions. Set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes. CMMI currently addresses three areas of interest:

(1) Product and service development --- CMMI for development (CMMI DEV),

(2) Service establishment, management, and delivery --- CMMI for Services (CMMI SVC), and

(3) Product and service acquisition ---- CMMI for Acquisition (CMMI ACQ). CMMI was developed by a group of experts from industry, government, and the Software Engineering Institute (SEI) at carnage Mellon University. CMMI models provide guidance for developing or improving processes that meet the business goals

of an organization. A CMMI model may also be used as a framework for appraising the process maturity of the organization. Depending on the CMMI constellation (acquisition, services, development), you use, the process areas it contains will vary. Key process areas are the areas that will be covered by covered by the organization’s processes. The table below lists the process areas that are present in all CMMI constellations. This collection of sixteen process areas is called the CMMI Model framework, or CMF.

CMMI Model Framework (CMF)

Capability Maturity model Integration (CMMI) Model framework (CMF) Abbreviation	**Name**	**Area**	**Maturity Level**
REQM	Requirements Management	Engineering	2
PMC	Project Monitoring and Control	Project Management	2
PP	Project Monitoring and Control	Project Management	2
CM	Configuration Management	Support	2
MA	Measurement and Analysis	Support	2
PPQA	Process and product Quality Assurance	Support	2

Process metrics

Software development process metrics can fall into one of the following categories:

- ■ Software process quality metrics
- ■ Software process timetable metrics
- ■ Error removal effectiveness metrics
- ■ Software process productivity metrics.

Software process quality metrics

Software process quality metrics may be classified into two classes:

■ Error density metrics

■ Error severity metrics.

Another group of indirect metrics that relates to software process quality is the McCabe’s cyclomatic complexity metrics .

Software quality metrics

Error density metrics

This section describes six different types of metrics. Calculation of error density metrics involves two measures:

(1) software volume, and

(2) errors counted.

Software volume measures. Some density metrics use the number of lines of code while others apply function points.

Errors counted measures. Some relate to the number of errors and others to the weighted number of errors. Weighted measures that ascertain the severity of the errors are considered to provide more accurate evaluation of the error situation. A common method applied to arrive at this measure is classification of the detected errors into severity classes, followed by weighting each class. The weighted error measure is computed by summing up multiples of the number of errors found in each severity class by the adequate relative severity weight.

Error seveity metrics

The metrics belonging to this group are used to detect adverse situations of increasing numbers of severe errors in situations where errors and weighted errors, as measured by error density metrics, are generally decreasing.

Software process timetable metrics

Software process timetable metrics may be based on accounts of success (completion of milestones per schedule) in addition to failure events (noncompletion per schedule). An alternative approach calculates the average delay in completion of milestones.

Error removal effectiveness metrics

Software developers can measure the effectiveness of error removal by the software quality assurance system after a period of regular operation usually 6 or 12 months) of the system. The metrics combine the error records of the development stage with the failures records compiled during the first year (or any defined period) of regular operation.

Software process productivity metrics

This group of metrics includes "direct" metrics that deal with a project's human resources productivity as well as "indirect" metrics that focus on the extent of software reuse. Software reuse substantially affects productivity and effectiveness.

An additional term – "benchmarking software development productivity" – has recently entered the list of metrics used to measure software process productivity (see Maxwell, 2001; Symons, 2001).

Product metrics

Product metrics refer to the software system's operational phase – years of regular use of the software system by customers, whether "internal" or "external" customers, who either purchased the software system or contracted for its development. In most cases, the software developer is required to provide customer service during the software's operational phase.

Customer services are of two main types:

■ **Help desk services (HD)** – software support by instructing customers regarding the method of application of the software and solution of customer implementation problems. Demand for these services depends to a great extent on the quality of the user interface (its "user friendliness") as well as the quality of the user manual and integrated help menus.

■ **Corrective maintenance services** – correction of software failures identified by customers/users or detected by the customer service team prior to their discovery by customers. The number of software failures and their density are directly related to software development quality. For completeness of information and better control of failure correction, it is recommended that all software failures detected by the customer service team be recorded as corrective maintenance calls.

Commonly, all customer services – namely, HD and corrective maintenance services – are provided to customers/users by a software support center (the "customer service center", among the many titles given to this service). It is expected that very few customer calls will be related to identified failures. In other words, most of the software support center's customer calls will be "nonfailure" calls. For those calls that deal with an identified failure and for cases where the maintenance team has detected a failure, a failure report is expected.

HD metrics are based on all customer calls while corrective maintenance metrics are based on failure reports. Product metrics generally rely on performance records compiled during one year (or any other specified period of time). This policy enables comparisons of successive years in addition to comparisons between different units and software systems.

Product metrics

The array of software product metrics presented here is classified as follows:

■ HD quality metrics

HD productivity and effectiveness metrics

■ Corrective maintenance quality metrics

■ Corrective maintenance productivity and effectiveness metrics.

It should be remembered that software maintenance activities include:

■ Corrective maintenance – correction of software failures detected during regular operation of the software.

■ Adaptive maintenance – adaptation of existing software to new customers
or new requirements.

■ Functional improvement maintenance – addition of new functions to the existing software, improvement of reliability, etc.

In the metrics presented here we limit our selection to those that deal with corrective maintenance. For other components of software maintenance, the metrics suggested for the software development process (*process metrics*) can be used as is or with minor adaptations.

HD quality metrics

The types of HD quality metrics discussed here deal with:

■ HD calls density metrics – the extent of customer requests for HD services as measured by the number of calls.

■ Metrics of the severity of the HD issues raised.

■ HD success metrics – the level of success in responding to these calls. A success is achieved by completing the required service within the time determined in the service contract.

HD calls density metrics

This section describes six different types of metrics. Some relate to the number of the errors and others to a weighted number of errors. As for size/volume measures of the software, some use number of lines of code while others apply function points. The sources of data for these and the other metrics in this group are HD reports.

Severity of HD calls metrics

The metrics belonging to this group of measures aim at detecting one type of adverse situation: increasingly severe HD calls. The computed results may contribute to improvements in all or parts of the user interface (its “user friendliness”) as well as the user manual and integrated help menus.

This metric, the **Average Severity of HD Calls (ASHC)**,
refers to failures detected during a period of one year (or any portion thereof,
as appropriate):

Success of the HD services

The most common metric for the success of HD services is the capacity to solve problems raised by customer calls within the time determined in the service contract (*availability*). Thus, the metric for success of HD services compares the actual with the designated time for provision of these services. For example, the availability of help desk (HD) services for an inventory management software package is defined as follows:

■ The HD service undertakes to solve any HD call within one hour.

■ The probability that HD call solution time exceeds one hour will not exceed 2%.

■ The probability that HD call solution time exceeds four working hours will not exceed 0.5%.

Product metrics

HD productivity and effectiveness metrics

Productivity metrics relate to the total of resources invested during a specified period, while effectiveness metrics relate to the resources invested in responding to a HD customer call.

HD productivity metrics

HD productivity metrics makes use of the easy-to-apply KLMC measure of maintained software system's size or according to function point evaluation of the software system.

HD effectiveness metrics

The metrics in this group refer to the resources invested in responding to customers' HD calls.

Corrective maintenance quality metrics

Software corrective maintenance metrics deal with several aspects of the quality of maintenance services. A distinction is needed between software system failures treated by the maintenance teams and failures of the maintenance service that refer to cases where the maintenance failed to provide a repair that meets the designated standards or contract requirements. Thus, software maintenance metrics are classified as follows:

■ **Software system failures density metrics** – deal with the extent of demand for corrective maintenance, based on the records of failures identified during regular operation of the software system.

■ **Software system failures severity metrics** – deal with the severity of software system failures attended to by the corrective maintenance team.

■ **Failures of maintenance services metrics** – deal with cases where maintenance services were unable to complete the failure correction on time or that the correction performed failed.

■ **Software system availability metrics** – deal with the extent of disturbances caused to the customer as realized by periods of time where the services of the software system are unavailable or only partly available.

Software system failures density metrics

The software system failures density metrics presented here relate to the number and/or weighted number of failures. The size of the maintenance tasks is measured by the total number of code lines of the maintained software as well as by the function point evaluation. The sources of data for these metrics are software maintenance reports.

Software system failures severity metrics

Metrics of this group detect adverse situations of increasingly severe failures in the maintained software. Results may trigger retesting of all or parts of the software system. The events measured relate either to the disturbances and damages caused to the customer (representing the customer's point of view) or to the resources required to resolve the failure (representing the interests of the maintenance team). The metric presented here can be used for both purposes, that is, to apply weights that refer to the severity of the disturbances and damages experienced by the customer, or to the extent of resources required by the maintainer. This metric, the **Average Severity of Software System Failures (ASSSF),** refers to software failures detected during a period of one year (or alternatively a half or a quarter of a year, as appropriate):

Failures of maintenance services metrics

As mentioned above, maintenance services can fail either because they were unable to complete the failure correction on time or when the correction performed failed and a repeated correction is required. The metrics presented here relate to the second type of maintenance failure. A customer call related to a software failure problem that was supposed to be solved after a previous call is commonly treated as a maintenance service failure. For practical purposes, many organizations limit the time frame for the repeat calls to three months, although the period can vary by type of failure or some other organizational criterion.

Software system availability metrics

User metrics distinguish between:

■ Full availability – where all software system functions perform properly

■ Vital availability – where no vital functions fail (but non-vital functions may fail)

■ Total unavailability – where all software system functions fail.

The source for all availability metrics is user failure records. The latter specify the extent of damage (non-vital failure, vital failure and total system failure) as well as duration (hours) for each failure

ISO 9000-3 quality management system: guiding principles

Eight principles guide the new ISO 9000-3 standard; these were originally set down in the ISO 9000:2000 standard (ISO, 2000b), as follows:

(1) **Customer focus**. Organizations depend on their customers and therefore should understand current and future customer needs.

(2) **Leadership.** Leaders establish the organization's vision. They should create and maintain an internal environment in which people can become fully involved in achieving the organization's objectives via the designated route.

(3) **Involvement of people.** People are the essence of an organization; their full involvement, at all levels of the organization, enables their abilities to be applied for the organization's benefit.
(4) **Process approach**. A desired result is achieved more efficiently when activities and resources are managed as a process.
(5) **System approach to management**. Identifying, understanding and managing processes, if viewed as a system, contributes to the organization's effectiveness and efficiency.
(6) **Continual improvement**. Ongoing improvement of overall performance should be high on the organization's agenda.
(7) **Factual approach to decision making.** Effective decisions are based on the analysis of information.
(8) **Mutually supportive supplier relationships.** An organization and its suppliers are interdependent; a mutually supportive relationship enhances the ability of both to create added value.

Quality management standards

ISO 9000-3: requirements

The current standard edition of ISO, 9000-3 (ISO 1997) includes 20 requirements that relate to the various aspects of software quality management systems.
The new ISO 9000-3 (ISO/IEC, 2001) offers a new structure, with its 22 requirements classified into the following five groups:
- ■ Quality management system
- ■ Management responsibilities
- ■ Resource management
- ■ Product realization
- ■ Management, analysis and improvement.

The new structure realizes a change in emphasis among the various subjects that make up the requirements, a totally new classification of SQA topics into standard sections and revision of requirement section titles. These changes reflect a gradual rather than a radical change of concepts as presented in the updated guiding principles.

IEEE Std 1012 – verification and validation

The IEEE Std 1012-1998 (IEEE, 1998) deals with the processes for determining whether a software product conforms to its requirements specifications (verification) and whether it satisfies the objectives of its intended use (validation). The standard adopts a broad range of applications, as demanded by the variety of verification and validation (V&V) methods available for use throughout the software life cycle. In response to developments in the field, the current standard has been substantially expanded from the 1986 version.

Purpose

The purposes of IEEE 1012-1998 are:

■ To establish a common framework for V&V activities and tasks for all software life cycle processes
■ To define V&V requirements, including their inputs and outputs
■ To define software integrity levels and the V&V tasks appropriate for each
■ To define the content of a SVVP (Software V&V Plan) document.

Underlying concepts

The concepts expressed in IEEE 1012-1998 respond to 10 basic issues:
(1) **Broad definition of V&V activities**. This enables the standard to embrace all the checking and investigative activities performed throughout the software life cycle: review, testing, method evaluation, hazard identification and risk analysis, among others.
SQA project process standards – IEEE software engineering standards
(2) **Software integrity levels and their V&V requirements**. The standard defines four integrity levels according to the criticality of a software function, module or unit, as follows:
■ "High" – a function that affects critical system performance
■ "Major" – a function that affects important system performance
■ "Moderate" – a function that affects system performance; however, availability of an alternative method of operation enables the system to overcome the associated difficulties ■ "Low" – a function that affects system performance only by inconveniencing the user.
IEEE 1012–1998 grades V&V requirements according to the integrity level;
The standard also requires that when preparing the software verification and validation plan (SVVP), integrity levels be assigned to each component of the product.
(3) **Prescriptive requirements**. IEEE Std 1012–1998 is a prescriptive standard in that it lists the tasks that shall be performed in the course of every activity initiated during the software life cycle. For each of these tasks, the standard provides the following information:
■ Detailed description of the performance methodology
■ Required inputs
■ Required outputs

■ Definition of integrity levels for which performance of the task is not mandatory
■ Optional V&V tasks to be performed during selected life cycle process.
The software life cycle architecture presented in the standard is structured as a three-level tree composed of:
(1) Processes
(2) Activities
(3) Tasks.
The six processes covered by the standard are:
(1) Management
(2) Acquisition
(3) Supply
(4) Development
(5) operation
(6) maintenance

IEEE Std 1028 – reviews

IEEE Std 1028-1997 (IEEE, 1997) limits itself to the technical issue of "how to perform a systematic review". According to the standard, a systematic review is defined as a review performed by a team according to a documented procedure that produces documented results. Methodological issues, such as when to carry out a review or what type of review is most appropriate, are sidestepped, to be determined by other standards or by project management.
The five types of systematic reviews covered are:
■ Management reviews
■ Technical reviews (referred to as "formal design reviews" in this book)
■ Inspections
■ Walkthroughs
■ Audits.

Purpose

The purpose of IEEE Std 1028-1997 is to define systematic review procedures that:
■ Are applicable for reviews performed throughout the software life cycle
■ Conform with the review requirements defined by other standards.

Underlying concepts

Three underlying concepts characterize the standard:
(1) **High formality**. The standard's high formality is manifested throughout, especially by requirements for authorization and documentation.
(2) **Follow-up**. The standard demands incorporation of follow-up and performance approval for corrections made in all its review activities.
(3) **Compliance with international and IEEE standards**. IEEE Std 1028-1997 complies with other IEEE standards and international standards (e.g., ISO/IEC 9000-3) that prescribe performance of reviews).

Especially noteworthy are IEEE/EIA Std 12207.0-1996, ISO/IEC 12207:1995 and IEEE Std 1012-1998.

The standard's content

The main portion of IEEE Std 1028-1997 entails:

■ Detailed definition of review requirements

■ An appendix that shows the standard's relationships to life cycle processes described in IEEE 730-1989, IEEE 1012-1998, IEEE 1074-1995 and ISO/IEC 12207:1995.

The standard devotes one chapter each (Chapters 4–8) to five types of reviews. It also applies the identical nine-component structure to all the requirements of the various review types, although the number of components varies according to the review's characteristics. For instance, the last two components of this structure, namely "data collection recommendation" and "improvement", are mentioned only in the chapters dealing with inspections and walkthroughs.

The components of the review requirement are documented in the following structure:

(1) **Introduction**

■ Purposes of each type of review

■ Typical examples of each type of software product.

(2) **Responsibilities**. The responsibilities section deals with participants in the review and the role of each. The standard provides a list of participants, some of whom are mandatory and others optional. For example, the mandatory participants of a technical review are decision-maker, review leader, recorder and technical staff. Optional participants are management staff, other team members and customer or user representatives. No optional participants are listed for inspections and walkthroughs, as these are peer reviews, or for audits, as this type of review depends solely on the auditors' professional qualifications.

(3) **Input**. This section deals with data inputs. Data are divided into mandatory and optional items, and vary with review type. The mandatory data items are common to all review types, as might be expected. They focus on a statement of the review's objectives and the software products to be reviewed.

For example, the data items mandatory for a walkthrough are:

■ A statement of objectives

■ The software product being examined

■ Standards in effect for the acquisition, supply, development, operation, and/or maintenance of the software product.

Optional data items are:

■ Regulations, standards, guidelines, plans and procedures against

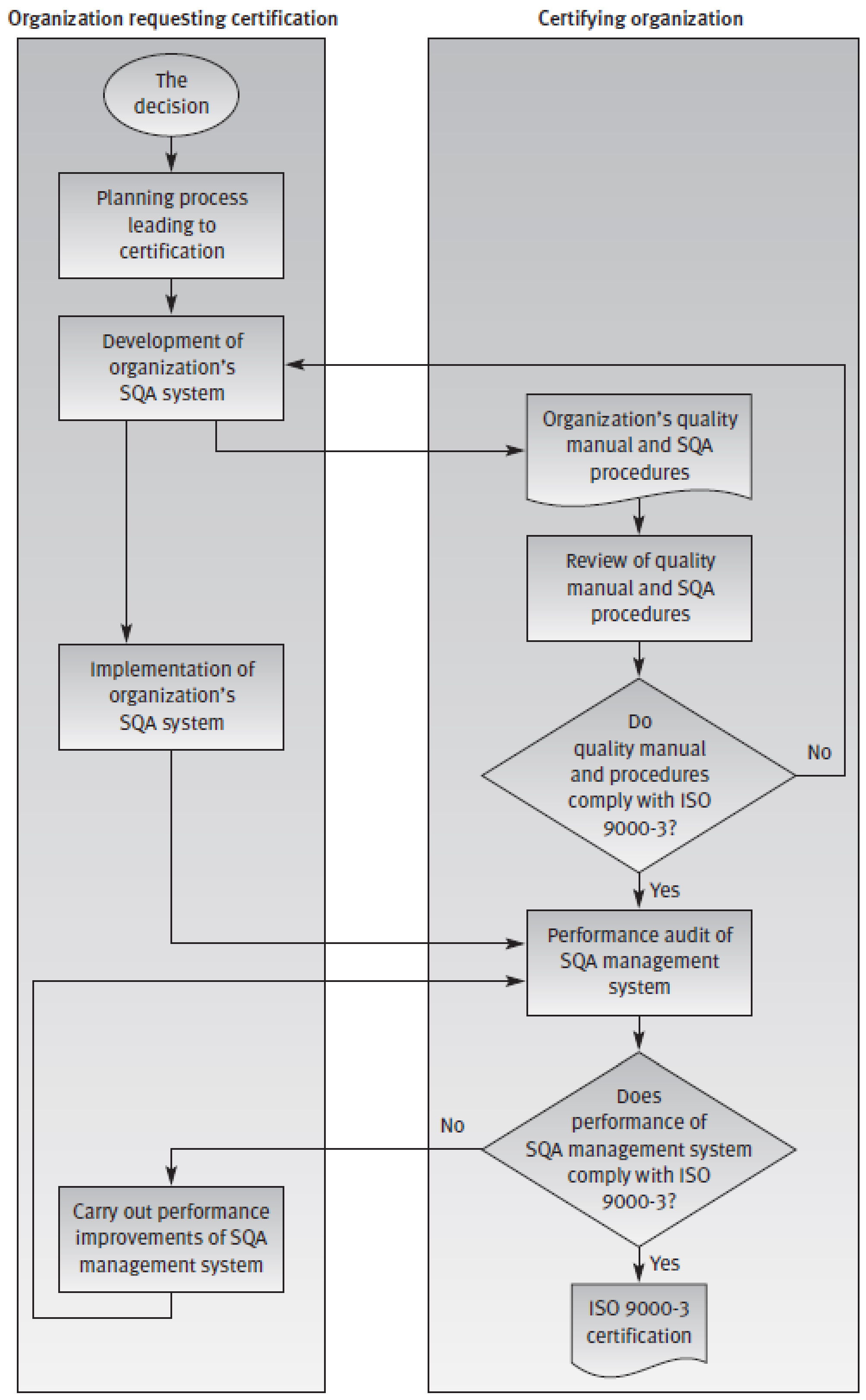
(4) **Entry criteria**. The review's authorization and performance preconditions represent what are otherwise known as entry criteria. Criteria common to all review types entail:

■ A statement of the review's objectives

■ Availability of the required input data.

(5) **Procedure**. Review procedures are required to include:
■ Management preparations
■ Planning of the review
■ Preparation by team members
■ Examination of the software product, including determination of required reworked software and corrections
■ Follow-up of performance of corrective activities.
(6) **Exit criteria**. The exit criteria specify what must be accomplished before the review can be officially concluded. These criteria include:
■ Completion of procedural activities
■ Follow-up and approval of satisfactory completion of action items or corrective and preventive actions
■ Completion of required review documentation.
(7) **Output**. The standard specifies the mandatory output items for each type of review. Additional items may be required by the organization, other local procedures, or specific cases.
(8) **Data collection recommendations**. It is recommended that inspection and walkthrough teams collect data related to anomalies encountered, where each case is classified and ranked according to its severity. This data will then be used to study the effectiveness and efficiency of current practices; they are also expected to stimulate improvements of methods and procedures.
(9) **Improvements**. The accumulated inspection and walkthrough data shall be analyzed in order to:
■ Formulate improved procedures
■ Update checklists used by the participants

■ Improve software development processes. which the software product is to be examined


Organization requesting certification
Certifying organization
The decision
Planning process leading to certification
Development of organization's SQA system
Organization's quality manual and SQA procedures
Review of quality manual and SQA procedures
Do quality manual and procedures comply with ISO 9000-3?
No
Yes
Implementation of organization's SQA system
Performance audit of SQA management system
Does performance of SQA management system comply with ISO 9000-3?
No
Yes
Carry out performance improvements of SQA management system
ISO 9000-3 certification

ISO 9000-3 certification process