Chapter 13: Software Quality

Th e Place of Software Quality in Project Planning

Quality will be of concern at all stages of project planning and execution, but will be of particular interest at the following points in the Step Wise framework.

- Step 1*: Identify project scope,* and objectives Some objectives could relate to the qualities of the application to be delivered.
- Step 2: *Identify project infrastructure,* Within this step, activity 2.2 identifies installation standards and procedures. Some of these will almost certainly be about quality.
- Step 3: *Analyse project characteristics,* In activity 3.2 ('Analyse other project characteristics – including quality based ones') the application to be implemented is examined to see if it has any special quality requirements. If, for example, it is safety critical then a range of activities could be added, such as n-version development where a number of teams develop versions of the same software which are then run in parallel with the outputs being cross-checked for discrepancies.
- Step 4: *Identify the products and activities of the project,* It is at this point that the entry, exit and process requirements are identified for each activity. This is described later in this chapter.
- Step 8: *Review and publicize plan,* At this stage the overall quality aspects of the project plan are reviewed.

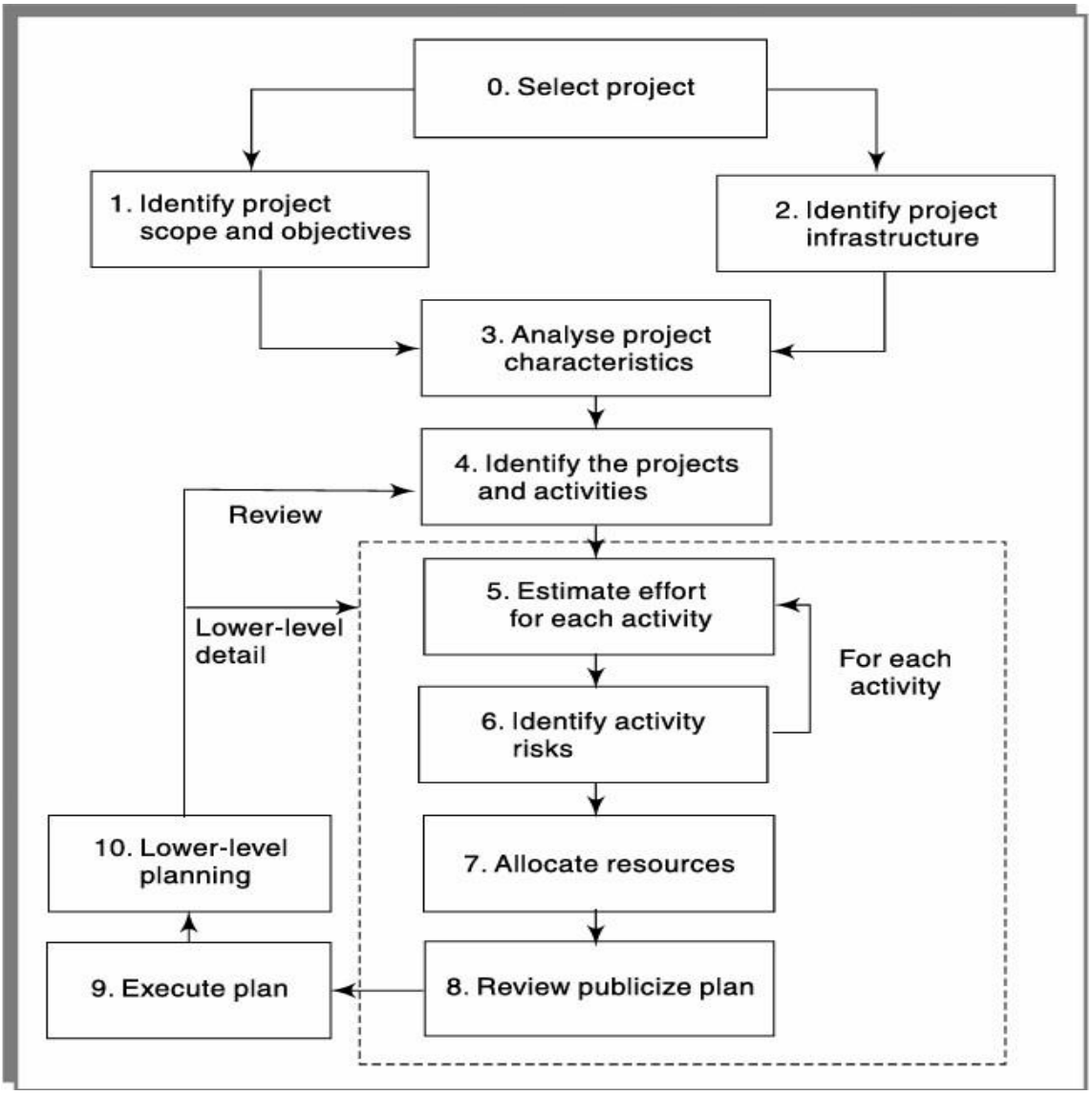


Fig: The place of so ware quality in Step Wise

The Importance of Software Quality

We would expect quality to be a concern of all producers of goods and services. However, the special characteristics of software create special demands.

- *Increasing criticality of software,* The final customer or user is naturally anxious about the general quality of software, especially its reliability. This is increasingly so as organizations rely more on their computer systems and software is used in more safety-critical applications, for example to control aircraft.
- *The intangibility of software,* can make it difficult to know that a project task was completed satisfactorily. Task outcomes can be made tangible by demanding that the developer produce 'deliverables' that can be examined for quality.
- *Accumulating errors during software development,* As computer system development comprises steps where the output from one step is the input to the next, the errors in the later deliverables will be added to those in the earlier steps, leading to an accumulating detrimental effect. In general, the later in a project that an error is found the more expensive it will be to fi x. In addition, because the number of errors in the system is unknown, the debugging phases of a project are particularly difficult to control. For these reasons quality management is an essential part of effective overall project management.

Defining Software Quality

Functional requirements defi ne what the system is to do, the resource requirements specify allowable costs and the quality requirements state how well this system is to operate.

Some qualities of a software product reflect the external view of software held by users, as in the case of usability. These external qualities have to be mapped to internal factors of which the developers would be aware. It could be argued, for example, that well-structured code is likely to have fewer errors and thus improve reliability.

Quality Measure

Defining quality is not enough. If we are to judge whether a system meets our requirements, we need to be able to measure its qualities. A good measure must relate the number of units to the maximum possible. The maximum number of faults in a program, for example, is related to the size of the program, so a measure of faults per thousand lines of code is more helpful than total faults in a program.

The measures may be direct, where we can measure the quality directly, or indirect, where the thing being measured is not the quality itself but an indicator that the quality is present. For example, the number of enquiries by users received by a help desk about how one operates a particular software application might be an indirect measurement of its usability.

For example, the number of errors found in program inspections could be counted, on the grounds that the more thorough the inspection process, the more errors will be discovered. This count could, of course, be improved by allowing more errors to go through to the inspection stage rather than eradicating them earlier – which is not quite the point.

Quality Specification

When there is concern about the need for a specific quality characteristic in a software product then a quality specification with the following minimum details should be drafted:

- definition/description: definition of the quality characteristic
- scale: the unit of measurement;
- test: the practical test of the extent to which the attribute quality exists;
- minimally acceptable: the worst value which might be acceptable if other characteristics compensated for it, and below which the product would have to be rejected out of hand;
- target range: the range of values within which it is planned the quality measurement value should lie;
  • now: the value that applies currently.

Reliability

The ability of system or component to perform its required functions under stated conditions for a specified period of time. There could be several measurements applicable to a quality characteristic. For example, in the case of reliability, this might be measured in terms of:

- availability: the percentage of a particular time interval that a system is usable;
- mean time between failures: the total service time divided by the number of failures;
- failure on demand: the probability that a system will not be available at the time required or the probability that a transaction will fail;
- support activity: the number of fault reports that are generated and processed.
- Maintainability: which is how quickly a fault, once detected, can be corrected.
- changeability, which is the ease with which the software can be modified.
- analysability, which is the ease with which causes of failure can be identified.

Software Quality Model:

The need for measure the quality of a software is always felt. Garvin's 8 dimensions of quality weren't developed specifically for the software system, they'll be applied once software system quality is taken into account.

1. Performance Quality: How well the product performs
2. Feature Quality: How well it supports the required features
3. Reliability: Probability of a product working satisfactorily within a specified period of time.
4. Conformance: Degree to which the product meets the requirements.
5. Durability: Measure of Product Life.
6. Serviceability: Will the software system be maintained or corrected in a tolerably short time period?
7. Aesthetics: The look and feel of the product.
8. Perception: Users opinion about the product quality.

McCall' Model

McCall defined the quality of a software in terms of three broad parameters:

1. It's operational characteristics
2. How easy it is to fix defects
3. How easy it is to port it to different platforms

The above 3 high level quality attributes are defined based on the following 11 attributes

1. Correctness: the functionality should match the specification
2. Reliability: The extent to which the product works satisfactorily

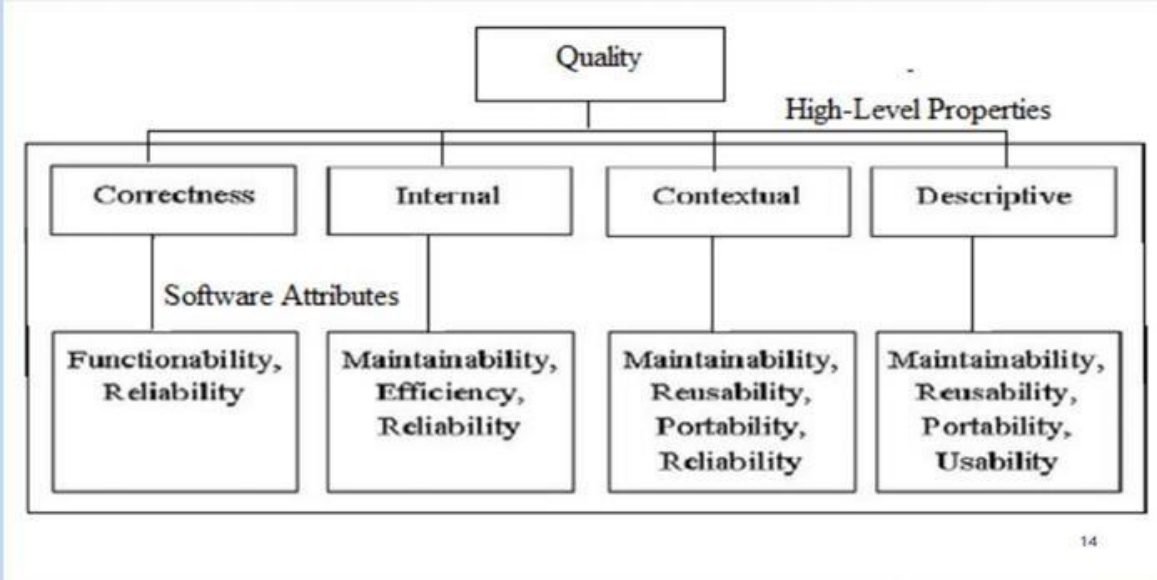
3. Efficiency: the amount of computing resources required for working
4. Integrity: the extent to which software data remains valid
5. Usability: the software should be easy to use
6. Maintainability: the ease with which it is possible to locate the fixed bugs.
7. Flexibility: the ability to make changes in the software product according to the business demands
8. Testability: the effort required to test software product to ensure that it performs its intended function

Product transition

- Portability: the effort required to transfer a software from one environment to other environment
- Reusability: the extent to which a software can be reused in other applications
- Interoperability: the ease in which all the components work with each other

Dormey's Model

Dormey proposed that software quality depends on four major high-level properties of the software correctness, internal characteristics. Contextual, descriptive. Each of these in turn depends on lower level software attributes

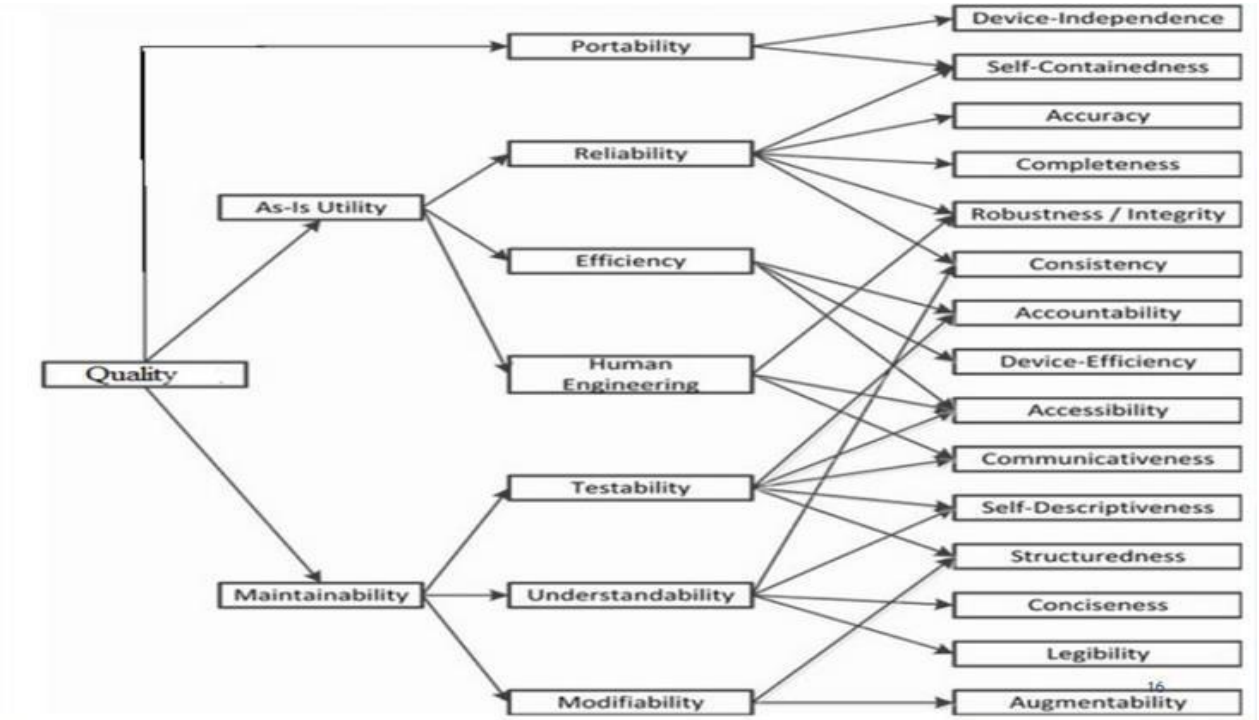


Boehm's Model

Boehm software quality model is used to represent hierarchical model the structure around high-level characteristics, intermediate level characteristics and primitive characteristics.

The model defends the quality of software on the basis of a set of credential and measurements the high level of characteristics is made in such a way that answers following questions:

- As-Is Utility: how well (easily, reliably and efficiently) can it be used.
- Maintainability: this aspect decides how convenient it is to understand, change and then re-test the process
- Portability: this aspect helps in deciding an effective way to change an environment



ISO 9126

Stands for "International Organization for Standardization. This standard was introduced in 1991 to tackle the question of definition of Software Quality.

Software Quality model recognizes 6 major external software characteristics:

- Functionality: covers the functions that the software product provides to satisfy user needs. • Reliability: capability of software to maintain its level of performance
- Usability: which refers to effort needed to use the software.
- Efficiency: which relates to the physical resource used when the software is executed.
- Maintainability: relates to the effort needed to make changes to the software
- Portability: relates to the ability of the software to be transferred to a different environment These characteristics are broken down into sub characteristics

ISO 9126-1 internal/external quality model

Characteristics	Subcharacteristcs
Functionality	suitability
	accuracy
	interoperability
	security
	functionality compliance
Reliability	maturity
	fault tolerance
	recoverability
	reliability compliance
Usability	understandability
	learnability
	operability
	attractiveness
	usability compliance
Efficiency	time behaviour
	resource utilisation
	efficiency compliance
Maintainability	analysability
	changeability
	stability
	testability
	maintainability compliance
Portability	adaptability
	installability
	co-existence
	replaceability
	portability compliance

Sub- characteristics

Functional Compliance: Refers to the degree to which software adheres to application related status or legal requirements

Maturity: the more the software has been used the more faults will have been uncovered and removed

Attractiveness: applicable to software products like games and other entertainment products

Learnability: easy to learn

Stability: low risk of modification

Replaceability: compatibility between old software components and new ones

Coexistence: ability of software to share resources with other software components

Product and process Metrics

Product metrics- describe the characteristics of the product such as size, complexity, design, features, performance, and quality level

Measuring the size: software sign can be described with three attributes-

- Length: it is the physical size of the product (LOC)
- Functionality: it describes the function supplied by the product to the user (Functional Points)
- Effort: person months required to build the project.

- Complexity: algorithmic complexity- measures the complexity of the algorithm implemented to solve the problem.

Process matrics: these characteristics can be used to improve the development and maintenance activities of the software process matrix helps measure how a development process is performing

Examples of process matrics are:

- average number of defects inspection hour
- average defect correction time

Product versus process quality management

- The system development process is made up of a number of activities that are linked together so that the output from one activity is the input to the next activity.
- Errors can enter the process at any stage. They can be caused either when programmers make mistakes in the logic of their programme or because information has not been passed clearly between the stages.
- Errors that creep in at the early stages are more expensive to correct at later stages, over the following reasons
- The later the error is found the more river cut more stages of development will be needed. If an error in the specification is found at the testing stage, then this will mean rework cut all the stages between specification and testing
- Each successive stage of development is more detailed and less able to absorb change.

Product versus Process Quality Management

Error should therefore be eradicated by careful examination of the products of each stage before they are passed out to the next. To do this, the following process requirements should be specified for each activity

- Entry requirements: which have to be placed before an activity can start. an example would be that a compulsive set of test data and expected result be prepared and approved before programme testing can commence
- Implementation requirements: which define how the process is to be conducted. In the testing phase for example, it might be laid down that whenever an error is found and corrected all test runs must be repeated even those that have previously been found to run correctly.
- Exit requirements: which have to be fulfilled before an activity is deemed to have been completed. For example, for the testing phase to be recognised as being completed, all tests will have to have been run successfully with no outstanding errors.

Quality management system BS EN ISO 9001:2000 QMS Requirements

The British standard for quality management is called BS EN 9000 (2000) which is identical to international standard ISO 9000: 2000

The standard identifies the eight quality management principles that can be used by senior managers to leave the organisation towards improved performance. The principles are:

- Customer focus
- Involvement of people
- System approach to management
- Factual approach to decision making
- Leadership
- Process approach

- Continual improvement
- Mutually beneficial supplier relationship

Process capability models

Every production process is subject to variations that limit our ability to produce a defect free. Product process capability models (PCMs) are used to qualify likely process variation which can then be included during the analysis of a product design the following process capabilities models:

- SEI CMM
- CMMI

SEI CMM

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of software development organisation. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process. It classifies organisations into five maturity levels.

Level 1: Initial

Ad hoc activities characterize a software development organization at this level. Very few are no processes are described at followed. Since software production processes are not limited, different engineers follow their process and as a result development efforts become chaotic. Therefore, it is also called a chaotic level.

Level 2: Repeatable

At this level, the fundamental project management practices like tracking cost and schedule are established. Size and cost estimation methods, like function point analysis, COCOMO etc are used.

Level 3: Defined

At this level, the methods for both management and development activities are defined and documented. There is a common organisation-wide understanding of operation, roles and responsibilities. The ways though defined, the process and product qualities are not measured.

Level 4: Managed

At this level the focus is on software metrics. Two kinds of metrics are composed product metrics measure the features of the product being developed, such as its size reliability time complexity understandability etc. Process metrics follows the effectiveness of the process being used, productivity, the average number of defects found for her inspection the average number of failure detected during testing for LOC, etc

Level 5: Optimizing

At this phase, process and product metrics are collected. Process and product measurement dates are evaluated for continuous process improvement.

CMMI (Capability Maturity Model Integration)

The CMMI model breaks down organisational majority into five levels. For businesses that embrace CMMI, the goal is to raise the organisation up to level 5 the optimising maturity level CMMI's five maturity levels are

- Initial: Processors are viewed as unpredictable and reactive. At this stage, “work gets completed but it’s often delayed and over budget.” This is the worst stage a business you can find itself in- an unpredictable environment that increases risk and inefficiency.
- Managed: There is a level of project management achieved. Projects are “planned, performed, measured and controlled” at this level. But there are still a lot of issues to address.
- Defined: at this stage, organisations are more proactive than reactive. There’s a set of “organisation-wide standards” to “provide guidance across projects programmes and portfolios.” Businesses understand their shortcomings how to address them and what the goal is for improvement.
- Quantitatively managed: This stage is more measured and controlled. The organisation is working off quantitative data to determine predictable process that align with stakeholder needs. The business is ahead of risk, with more data-driven insights into process deficiencies.
- Optimizing: Here, an organization’s processes are stable and flexible. At this final stage an organisation will be in constant state of improving and responding to changes or other opportunities. The organisation is stable, which allows for more “agile and innovation,” in predictable environment.

Chapter 9

Review

From a manager’s perspective, review of work products is an important mechanism for monitoring the progress of a project and ensuring the quality of the work products. Every project is developed through iterations over a large number of work products such as requirements document, design document, project plan document, code, etc. Each of these work products can have a large number of defects in them due to mistakes committed by the development team members. It is necessary to eliminate as many defects in these work products to realize a product of acceptable quality. Testing is an effective defect removal mechanism.

Utility of review

Besides being a cost-effective defect removal mechanism, review of any work product has several other benefits including the following mentioned below:

- Review usually helps to identify any deviation from standards, including issues that might affect maintenance of the software.
- Reviewers suggest ways to improve the work product such as using algorithms that are more time or space efficient, specific work simplifications, better technology opportunities that can be exploited, etc.
- In addition to defect identification, a review meeting often provides learning opportunities to not only the author of a work product, but also the other participants of the review meeting. The lessons acquired from a review meeting allows participants to avoid committing similar defects that were discussed in the review meeting and also allows them to make use of the best practices that were suggested.
- The review participants gain a good understanding of the work product under review, making it easier for them to interface or use the work product in their work

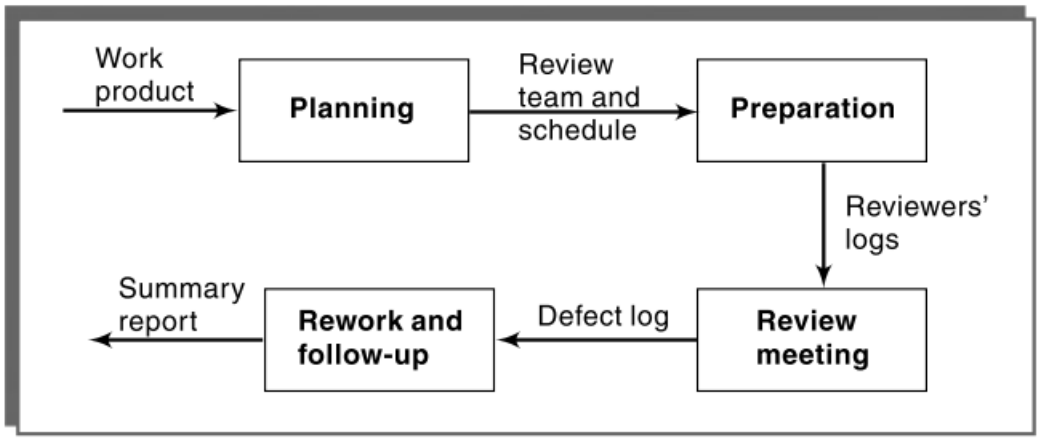
Review roles

In every review meeting, a few key roles need to be assigned to the review team members. These roles are moderator, recorder and the reviewers. The moderator plays a key role in the review process. The principal

responsibilities of the moderator include scheduling and convening meetings, distributing review materials, leading and moderating the review sessions, ensuring that the defects are tracked to closure. The main role of the recorder is to record the defects found, the time, and effort data. The review team members review the work product and give specific suggestions to the author about the existing defects and also point out ways to improve the work product.

Review process

Review of any work product consists of the following four important activities, viz. planning, review preparation and overview, review meeting, rework and follow-up. A review process model is shown in Figure 9.5. The model in Figure 9.5 captures the sequence of the activities that need to be carried out, the input to the activities, and the output produced from the activities. In the following, we briefly discuss these review activities.



Review process model

- **Planning** Once the author of a work product is ready for submitting the work for review; the project manager nominates a moderator. A moderator can be someone who is familiar with the work product. In consultation with the moderator, the project manager nominates the other members of the review team. Usually, the review process works best when the number of members is between five and seven.
- **Preparation** To initiate the review process, the moderator convenes a brief preparation meeting. In the preparation meeting, copies of the work product are distributed to the review team members. The author presents a brief overview of the work product. The moderator highlights the objectives of the review. The reviewers then individually carry out review and record their observations in separate documents called review logs.
- Review Meeting In the review meeting the reviewer's give their comments based on the logs they have prepared beforehand. The comments may pertain to a defect, work simplification, maintainability, etc. The author responds to the reviewers' comments and in this discussion other reviewers may also take part. The moderator ensures that the discussions remain focused and productive. The recorder scribes all the defects and points that the author agrees to, as well as the review statistics in the form of a review log.
- **Rework** The author addresses all the issues raised by the reviewers by carrying out the necessary modifications to the work product and prepares a rejoinder to all the points scribed in the review log. The rejoinder records the exact ways in which the comments have been taken care of by the author. The corrected work product along with the author's rejoinder is circulated among all the review team members. In a final brief meeting, the review team members check whether all the issues scribed in the review log have been resolved satisfactorily. At the end of this meeting, a final summary report of the review is prepared.

Data collection

Since a review meeting is a completely human endeavour, unless the data representing the results of the meetings is properly recorded, it can get lost. In addition to recording all defects, the data about the time spent by the reviewers in the review activity must also be captured. A record of the defect data is needed for tracking defects in the project.

The different reports in which the review data are captured are as follows:

1. **Review Preparation** Log Each reviewer prepares a review preparation log. The different items recorded in it by the reviewer are the data about defects he observes, their locations, their criticality, and the total time spent in doing the review of the work product.
2. **Review Log** In the review log only those defects that are agreed to by the author are logged. Defect logs are a crucial record since these help in tracking all defects to closure.
3. **Review Summary Report** This report summarizes the review data and presents an overall picture of the review. It contains information regarding the total defects and the amount of time spent on each of the review process activities.

Project Termination Review

A manager decides when a project should be terminated. As soon as a decision regarding project termination is taken, it is a good practice to conduct a project review meeting. Project termination reviews are important for successful, failed, as well as prematurely abandoned projects. The project termination review meeting marks the official closure of a project. Project termination reviews provide important opportunities to learn from past mistakes as well as successes. By analysing past mistakes the project teams can learn to do better by improving their methods and practices. The project termination review summary report is not only beneficial to the terminated project, but it can also benefit of other teams and therefore should be disseminated across the organization. It is important to note that project termination need not necessarily mean project failure or premature abandonment. A project may be terminated for a variety of reasons, including successful completion of the endeavour.

Reasons for project termination Here are a few reasons why a project gets terminated before the natural closing date:

- Project is completed successfully and handed over to the customer.
- Incomplete requirements
- Lack of resources
- Some key technologies used in the project have become obsolete during project execution.
- Economics of the project has changed, for example, because many competing products may have become available in the market.

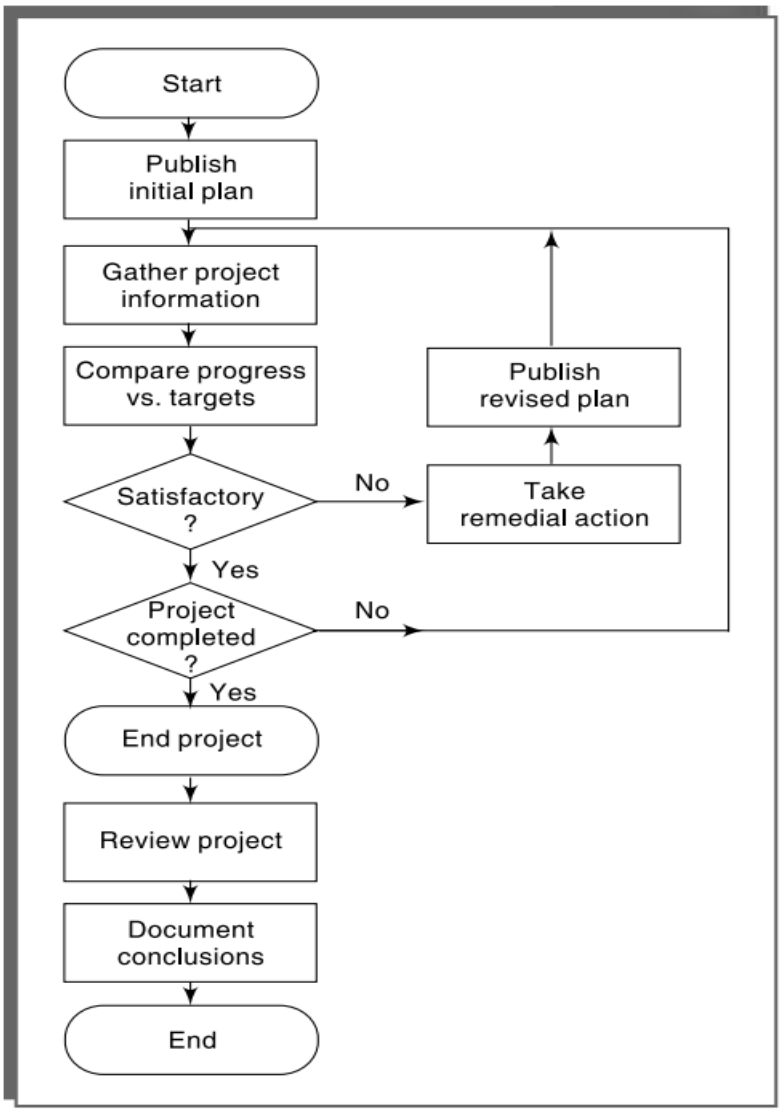
Project termination process

The important activities that are carried out as a part of the project termination review process are as follows:

- **Project Survey** The objective of the project survey activity is to collect various types of information pertaining to the project, without compromising the confidentiality of the respondents. An electronic survey is usually very effective. The information is collected through a set of carefully designed questionnaire that can bring out the important process and management issues, which have a strong bearing on the success or failure of the project.

- **Collection of Objective Information** A critical aspect of the postmortem review is to collect various project metrics. Real data helps to focus discussions on most crucial issues during the postmortem review. The different types of metrics that are collected include the cost, schedule, and quality metrics.
- **Debriefing Meeting** A debriefing meeting is a preparatory meeting that helps to ensure the fi nal project review meeting focuses on the most relevant aspects. In this meeting, only the senior members of the team participate. The debriefing meeting helps to obtain some direct feedback about the project from the senior members of the team.
- **Final Project Review** This meeting usually addresses various issues arising out of project planning and tracking, preliminary phases (requirements analysis, specification and design), configuration management, verification, and validation. Guided by the information collected in the previous steps, the project leaders determine the focus of the review discussions only on the relevant topics in various project activities. For example, if the collected data or the debriefing meeting data suggest schedule slippage, discussions on how the schedule slippage could have been avoided are conducted. It should be remembered that fault finding or blaming individuals must be avoided.
- **Result Publication** The project leader summarizes the positive and negative findings arrived at during the termination review process as well as prescriptions for improvement. The summary is published so that all the teams can to refer to it and also the management can take initiative for any necessary corrections based on it

Project control life cycle



: project control cycle