# UNIT-III

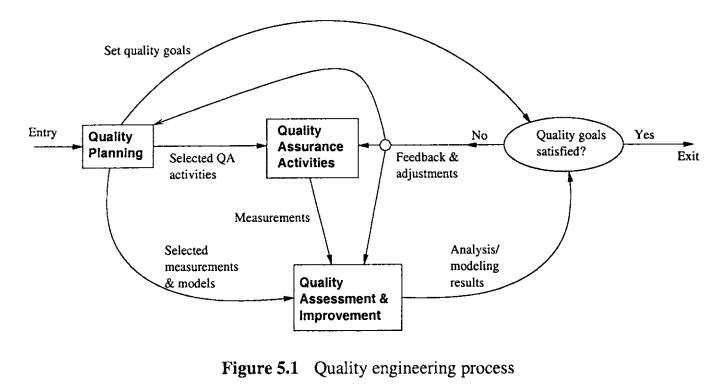
# QUALITY ENGINEERING

**Quality engineering: Activities and Processes**

**Quality Engineering:** Different customers and users have different quality expectations under different market environments. So, Instead of just doing QA activities, we have to do quality engineering. In quality engineering,our goal is to meet or exceed these quality expectations through the selection and execution of appropriate QA activities while minimizing the cost and other project risks under the project constraints.

There are three major groups of activities in the quality engineering process:

1. Pre-QA activities
2. In-QA activities
3. Post-QA activities



1. **Pre-QA activities:Quality planning.** These are the activities that should be carried out before carrying out the regular QA activities. There are two major types of pre-QA activities in quality planning, including:
   1. Set specific quality goals.
   2. Form an overall QA strategy, which includes two sub-activities:
      1. Select appropriate QA activities to perform.
      2. Choose appropriate quality measurements and models to provide feedback, quality assessment and improvement.
2. **In-QA activities: Executing planned QA activities and handling discovered defects.** In addition to performing selected QA activities, an important part of this normal execution is to deal with the discovered problems.
3. **Post-QA activities: Quality measurement, assessment and improvement.** These are the activities that are carried out after normal QA activities have started but not as part of these normal activities. The primary purpose of these activities is to provide quality assessment and feedback so that various management decisions can be made and possible quality improvement initiatives can be carried out.

**Quality engineering and QIP**

Quality improvement was achieved through measurement, analysis, feedback, and organizational support. The overall framework is called QIP, or quality improvement paradigm.

QIP includes three interconnected steps: understanding, assessing, and packaging, which form a feedback and improvement loop, as briefly described below:

1. The first step is to understand the baseline so that improvement opportunities can be identified and clear, measurable goals can be set. All future process changes are measured against this baseline.

2. The second step is to introduce process changes through experiments, pilot projects, assess their impact, and fine tune these process changes.

3. The last step is to package baseline data, experiment results, local experience, and updated process as the way to infuse the findings of the improvement program into the development organization. QIP and related work on measurement selection and organizational support are described further in connection to defect prevention and in connection to quality assessment and improvement.Our approach to quality engineering can be considered as an adaptation of QIP to assure and measure quality, and to manage quality expectations of target customers.

Some specific correspondences are noted below: 0 our pre-QA activities roughly correspond to the understand step in QIP. The execution of our selected QA strategies correspond to the “changes” introduced in the assess step in QIP. However, we are focusing on the execution of normal QA activities and the related measurement activities selected previously in our planning step, instead of specific changes.

Our analysis and feedback (or post-QA) activities overlap with both the assess and package steps in QIP, with the analysis part roughly corresponding to the QIP-assess step and the longer term feedback roughly corresponding to the QIP-package step.

**2. QUALITY PLANNING: GOAL SETTING AND STRATEGY FORMATION**

As mentioned above, pre-QA quality planning includes setting quality goals and forming a QA strategy.

The general steps include: 1. Setting quality goals by matching customer’s quality expectations with what can be economically achieved by the software development organizations in the following sub-steps: (a) Identify quality views and attributes meaningful to target customers and users. (b) Select direct quality measures that can be used to measure the selected quality (c) Quantify these quality measures to set quality goals while considering the marattributes from customer’s perspective. ket environment and the cost of achieving different quality goals.

2. In forming a QA strategy, we need to plan for its two basic elements: (a) Map the above quality views, attributes, and quantitative goals to select a specific set of QA alternatives. (b) Map the above external direct quality measures into internal indirect ones via selected quality models. This step selects indirect quality measures as well as usable models for quality assessment and analysis.

**Setting quality goals**

One important fact in managing customer’s quality expectations is that different quality attributes may have different levels of importance to different customers and users.Relevant quality views and attributes need to be identified first.

For example, reliability is typically the primary concern for various business and commercial software systems because of people’s reliance on such systems and the substantial financial loss if they are malfunctioning. Similarly, if a software is used in various real-time control situations, such as air traffic control software and embedded software in automobile, medical devices, etc., accidents due to failures may be catastrophic. Therefore, safety is the major concern.

On the other hand, for mass market software packages, such as various auxiliary utilities for personal computers, usability, instead of reliability or safety, is the primary concern. Even in the narrower interpretation of quality we adopted in this book to be the correctnesscentered quality attributes associated with errors, faults, failures, and accidents, there are different types of problems and defects that may mean different things to different customers. For example, for a software product that is intended for diverse operational environments, inter-operability problems may be a major concern to its customers and users; while the same problems may not be a major concern for software products with a standard operational environment. Therefore, specific quality expectations by the customers require us to identify relevant quality views and attributes prior to setting appropriate quality goals. This needs to be done in close consultation with the customers and users, or those who represents their interests, such as requirement analysts, marketing personnel, etc.

Once we obtained qualitative knowledge about customers’ quality expectations, we need to quantify these quality expectations to set appropriate quality goals in two steps:

1. We need to select or define the quality measurements and models commonly accepted by the customers and in the software engineering community. For example, reliability and safety are examples of correctness-centered quality measures that are meaningful to customers and users, which can be related to various internal measures of faults commonly used within software development organizations.

2. We need to find out the expected values or ranges of the corresponding quality measurements. For example, different market segments might have different reliability expectations. Such quality expectations are also influenced by the general market conditions and competitive pressure. Software vendors not only compete on quality alone, but also on cost, schedule, innovation, flexibility, overall user experience, and other features and properties as well. Zero defect is not an achievable goal under most circumstances, and should not be the goal. Instead, zero defection and positive flow of new customers and users based on quality expectation management should be a goal.

In a sense, this activity determines to a large extent the product positioning vs. competitors in the marketplace and potential customers and users. Another practical concern with the proper setting of quality goals is the cost associated with different levels of quality. This cost can be divided into two major components, the failure cost and the development cost. The customers typically care more about the total failure cost,Cf, which can be estimated by the average single failure cost, cf, and failure probability, pf, over a pre-defined duration of operation as:

Cf =cf xpf.

This failure probability can be expressed in terms of reliability, R, as pf = 1 - R, where R is defined to be the probability of failure-free operations for a specific period of given set of input. To minimize Cf, one can either try to minimize cf or pf. However, cf is typically determined by the nature of software applications and the overall environment the software is used in. Consequently, not much can be done about cf reduction without incurring substantial amount of other cost. One exception to this is in the safety critical systems, where much additional cost was incurred to establish barriers and containment in order to reduce failure impact. On the other hand, minimizing pf, or improving reliability, typically requires additional development cost, in the form of additional testing time, use of additional QA techniques, etc. Therefore, an engineering decision need to be made to match the quantified customer’s quality expectations above with their willingness to pay for the quality. Such quantitative cost-of-quality analyses should help us reach a set of quality goals.

**Forming a QA strategy**

Once specific quality goals were set, we can select appropriate QA alternatives as part of a QA strategy to achieve these goals. Several important factors need to be considered:

*The influence of quality perspectives and attributes*: For different kinds of customers, users, and market segments, different QA alternatives might be appropriate, because they focus on the assurance of quality attributes based on this specific perspective. For example, various usability testing techniques may be useful for ensuring the usability of a software product, but may not be effective for ensuring its functional correctness.

*The influence of different quality Levels:* Quantitative quality levels as specified in the quality goals may also affect the choice of appropriate QA techniques. For example, systems with various software fault tolerance features may incur substantially more additional cost than the ones without them. Therefore, they may be usable for highly dependable systems or safety critical systems, where large business operations and people’s lives may depend on the correct operations of software systems, but may not be suitable for less critical software systems that only provide non-essential information to the users.

Notice that in dealing with both of the above factors, we assume that there is a certain relationship between these factors and specific QA alternatives. Therefore, specific QA alternatives need to be selected to fulfill specific quality goals based on the quality perspectives and attributes of concern to the customers and users.

Implicitly assumed in this selection process is a good understanding of the advantages and disadvantages of different QA alternatives under different application environments. These comparative advantages and disadvantages are the other factors that also need to be considered in selecting different QA alternatives and related techniques and activities. These factors include cost, applicability to different environments, effectiveness in dealing with different kinds of problems. In order to achieve the quality goals, we also need to know where we are and how far away we are from the preset quality goals. To gain this knowledge, objective assessment using some quality models on collected data from the QA activities is necessary. There are direct quality measures and indirect quality measures. The direct quality measures need to be defined as part of the activities to set quality goals, when such goals are quantified.

Under many situations, direct quality measures cannot be obtained until it is already too late. For example, for safety critical systems, post-accident measurements provide a direct measure of safety. But due to the enormous damage associated with such accidents, we are trying to do everything to avoid such accidents. To control and monitor these safety assurance activities, various indirect measurements and indicators can be used. For all software systems there is also an increasing cost of fixing problems late instead of doing so early in general, because a hidden problem may lead to other related problems, and the longer it stays undiscovered in the system, the further removed it is from its root causes, thus making the discovery of it even more difficult. Therefore, there is a strong incentive for early indicators of quality that usually measure quality indirectly.

Indirect quality measures are those which can be used in various quality models to assess and predict quality, through their established relations to direct quality measures based on historical data or data from other sources. Therefore, we also need to choose appropriate measurements, both direct and indirect quality measurement, and models to provide quality assessment and feedback.

1. **QUALITY ASSESSMENT AND IMPROVEMENT**

Various parallel and post-QA activities are carried out to close the quality engineering loop. The primary purpose of these activities is to provide quality assessment and feedback so that various management decisions, such as product release, can be made and possible quality and process improvement initiatives can be carried out.

The major activities in this category include:

* *Measurement:* Besides defect measurements collected during defect handling, which is typically carried out as part of the normal QA activities, various other measurements are typically needed for us to track the QA activities as well as for project management and various other purposes. These measurements provide the data input to subsequent analysis and modeling activities that provide feedback and useful information to manage software project and quality.
* *Analysis and modeling:* These activities analyze measurement data from software projects and fit them to analytical models that provide quantitative assessment of selected quality characteristics or sub-characteristics. Such models can help us obtain an objective assessment of the current product quality, accurate prediction of the future quality, and some models can also help us identify problematic areas.
* *Providing feedback and identifying improvement potentials:* Results from the above analysis and modeling activities can provide feedback to the quality engineering process to help us make project scheduling, resource allocation, and other management decisions. When problematic areas are identified by related models, appropriate remedial actions can be applied for quality and process improvement.
* *Follow-up activities:* Besides the immediate use of analysis and modeling results described above, various follow-up activities can be carried out to affect the long term quality and organizational performance. For example, if major changes are suggested for the quality engineering process or the software development process, they typically need to wait until the current process is finished to avoid unnecessary disturbance and risk to the current project.

1. **QUALITY ENGINEERING IN SOFTWARE PROCESSES**

The quality engineering process forms an integral part of the overall software engineering process, where other concerns, such as cost and schedule, are also considered and managed. Individual QA activities can be carried out and integrated into the software process. When we broaden our scope to quality engineering, it also covers pre

QA quality planning as well as the post-QA measurement and analysis activities carried out parallel to and after QA activities to provide feedback and other useful information. All these activities and the quality engineering process can be integrated into the overall software process as well.

Pre-QA quality planning can be an integral part of any project planning. For example, in the waterfall process, this is typically carried out in the phase for market analysis, requirement gathering, and product specification. Such activities also provide us with valuable information about quality expectations by target customers and users in the specific market segment a software vendor is prepared to compete in. Quality goals can be planned and set accordingly. Project planning typically includes decisions on languages, tools, and technologies to be used for the intended software product. It should be expanded to include 1) choices of specific QA strategies and 2) measurement and models to be used for monitoring the project progress and for providing feedback.

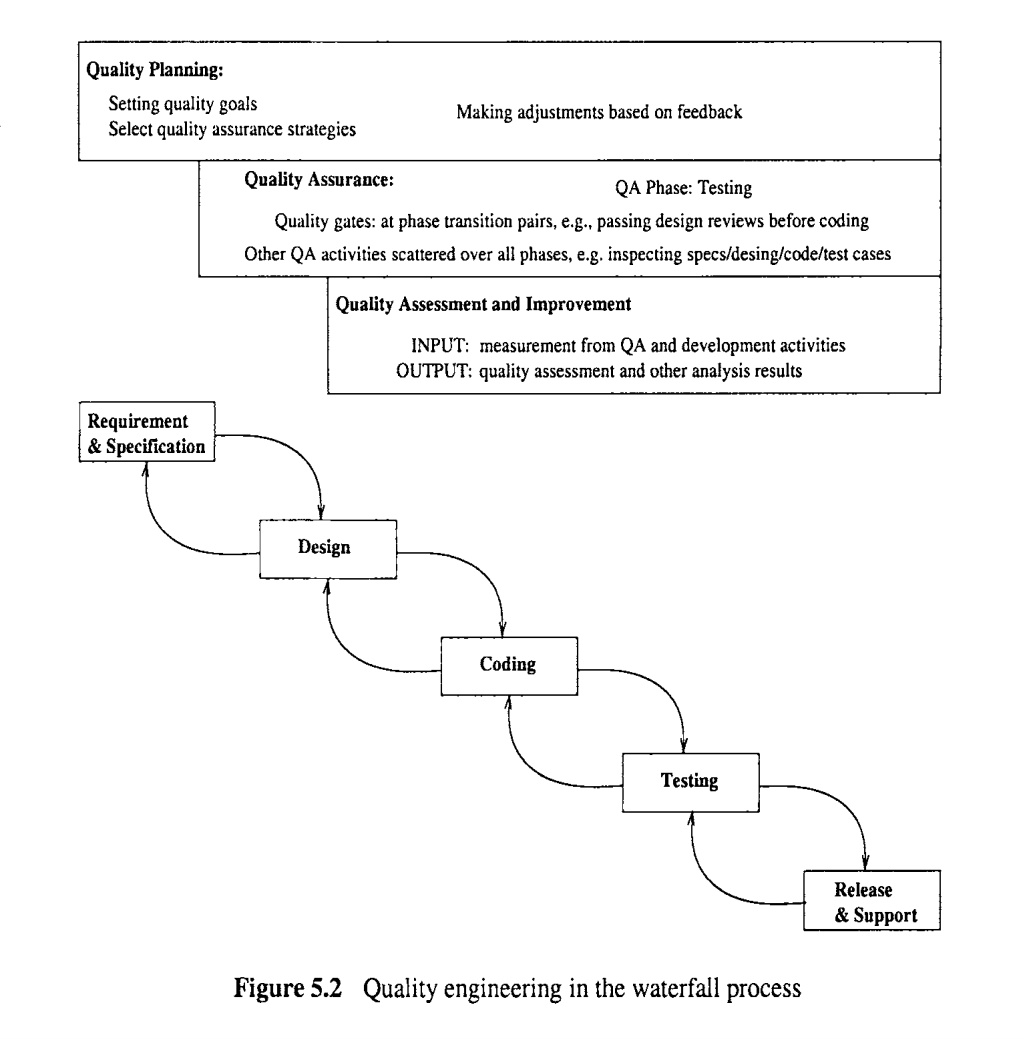
**In** alternative software processes other than waterfall, such as in incremental, iterative, spiral, and extreme programming processes, pre-QA activities play an even more active role, because they are not only carried out at the beginning of the whole project, but also at the beginning of each subpart or iteration due to the nature that each subpart includes more or less all the elements in the waterfall phases. Therefore, we need to set specific quality goals for each subpart, and choose appropriate QA activities, techniques, measurement, and models for each subpart. The overall quality goal may evolve from these sub-goals in an iterative fashion.

For normal project monitoring and management under any process, appropriate measurement activities need to be carried out to collect or extract data from the software process and related artifacts; analyses need to be performed on these data; and management decision can be made accordingly. On the one hand, the measurement activity cannot be carried out without the involvement of the software development team, either as part of the normal defect handling and project tracking activities, or as added activity to provide specific input to related analysis and modeling. Therefore, the measurement activities have to be handled “on-line” during the software development process, with some additional activities in information or measurement extraction carried out after the data collection and recording are completed.

On the other hand, much of the analysis and modeling activities could be done “off-line”, to minimize the possible disruption or disturbance to the normal software development process. However, timely feedback based on the results from such analyses and models is needed to make adjustments to the QA and to the development activities. Consequently, even such “off-line” activities need to be carried out in a timely fashion, but may be at a lower frequency. For example, in the implementation of testing tracking, measurement, reliability analysis, and feedback for IBM’s software products (Tian, 1996), dedicated quality analyst performed such analyses and modeling and provided weekly feedback to the testing team, while the data measurement and recording were carried out on a daily basis.

The specific analysis, feedback, and follow-up activities in the software quality engineering process fit well into the normal software management activities. Therefore, they can be considered as an integral part of software project management. Of course, the focus of these quality engineering activities is on the quality management, as compared to the overall project management that also includes managing project features, cost, schedule, and so on.

The integration of the quality engineering process into the waterfall software development process can be illustrated by Figure 5.2.



The horizontal activities roughly illustrate the timeline correspondence to software development activities. For example, quality planning starts right at the start of the requirement analysis phase, followed by the execution of the selected QA activities, and finally followed by the measurement and analysis activities. All these activities typically last over the whole development process, with different subactivities carried out in different phases. This is particularly true for the QA activities, with testing in the test phase, various reviews or inspections at the transition from one phase to its successor phase, and other QA activities scattered over other phases. Minor modifications are needed to integrate quality engineering activities into other development processes. However, the distribution of these activities and related effort is by no means uniform over the activities or over time, which is examined next.

**Effort profile**

Among the three major types of activities in the quality engineering process, the execution of specific QA activities is central to dealing with defects and assuring quality for the software products. Therefore, they should and normally do consume the most resources in terms of human effort as well as utilization of computing and other related resources. However, the effort distribution among the three is not constant over time because of the process characteristics described above and the shifting focus over time. Some key factors that affect and characterize the effort profile, or the effort distribution over time, include:

* Quality planning drives and should precede the other two groups of activities. Therefore, at the beginning part of product development, quality planning should be the dominant part of quality engineering activities. Thereafter, occasional adjustments to the quality goals and selected quality strategies might be applied, but only a small share of effort is needed.

The collective effort of selected QA activities generally demonstrates the following pattern:

- There is a gradual build-up process for individual QA activities and for them collectively.

- The collective effort normally peaks off a little bit before product release, when development activities wind down and testing activities are the dominant activities.

- Around product release and thereafter, the effort tapers off, typically with a sudden drop at product release.

Of course, the specific mix of selected QA activities as well as the specific development process used would affect the shape of this effort profile as well. But the general pattern is expected to hold. Measurement and quality assessment activities start after selected QA activities are well under way. Typically, at the early part of the development process, small amounts of such activities are carried out to monitor quality progress. But they are not expected to be used to make major management decisions such as product release. These activities peak off right before or at the product release, and lower gradually after that. In the overall shape and pattern, the effort profile for these activities follows that for the collective QA activities above, but with a time delay and a heavier load at the tail-end. One common adjustment to the above pattern is the time period after product release. Immediately after product release or after a time delay for market penetration, the initial wave of operational use by customers is typically accompanied by many user-reported problems, which include both legitimate failures and user errors. Consequently, there is typically an upswing of overall QA effort. New data and models are also called for, resulting in an upswing of measurement and analysis activities as well.

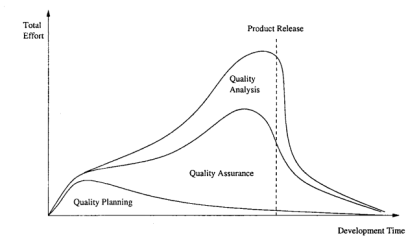
The main reason for this upswing is the difference between the environment where the product is tested under and the actual operational environment the product **is** subjected to.

The overall quality engineering effort over time is divided into three parts:

The bottom part represents the share of total effort by quality planning activities; The middle part represents the share of total effort for the execution of selected QA activities;

The upper part represents the share of total effort for the measurement and quality assessment activities.

Notice that this figure is for illustration purposes only. The exact profile based on real data would not be as smooth and would naturally show large amount of variability, with many small peaks and valleys. But the general shape and pattern should preserve.



In addition, the general shape and pattern of the profile such as in Figure 5.3 should preserve regardless of the specific development process used. Waterfall process would see more dominance of quality planning in the beginning, and dominance of testing near product release, and measurement and quality assessment activities peak right before product release.

Other development processes, such as incremental, iterative, spiral, and extreme programming processes, would be associated with curves that vary less between the peaks and valleys. QA is spread out more evenly in these processes than in the waterfall process, although it is still expected to peak a little bit before product release. Similarly, measurement and analysis activities are also spread out more evenly to monitor and assess each part or increment, with the cumulative modeling results used in product release decisions. There are also more adjustments and small-scale planning activities involved in quality planning, which also makes the corresponding profiles less variable as well.

To manage the quality assurance (QA) activities and to provide realistic opportunities of quantifiable quality improvement, we need to go beyond QA to perform the following:

*Quality planning* before specific QA activities are carried out, in the so-called pre-QA activities in software quality engineering. We need to set the overall quality goal by managing customer’s quality expectations under the project cost and budgetary constraints. We also need to select specific QA alternatives and techniques to implement as well as measurement and models to provide project monitoring and qualitative feedback.

*Qualily quantization and improvement* through measurement, analysis, feedback, and follow-up activities. These activities need to be carried out after the start of specific QA activities, in the so-called post-QA activities in software quality engineering.

The analyses would provide us with quantitative assessment of product quality, and identification of improvement opportunities. The follow-up actions would implement these quality and process improvement initiatives and help us achieve quantifiable quality improvement.