**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.

**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.