10

IT Project Quality Management

CHAPTER OVERVIEW

The focus of this chapter will be on several concepts and philosophies of quality management. By learning about the people who founded the quality movement over the last fifty years, we can better understand how to apply these philosophies and teachings to develop a project quality management plan. After studying this chapter, you should understand and be able to:

- Describe the Project Management Body of Knowledge (PMBOK) area called project quality management (PQM) and how it supports quality planning, qual ity assurance, quality control, and continuous improvement of the project's products and supporting processes.
- Identify several quality gurus, or foundersof the quality movement, and their role in shaping quality philosophies worldwide.
- Describe some of the more common quality initiatives and management sys tems that include ISO certification, Six Sigma, and the Capability Maturity Model (CMM) for software engineering.
- Distinguish between validation and verification activities and how these activi ties support IT project quality management.
- Describe the software engineering discipline called configuration management and how it is used to manage the changes associated with all of the project's deliverables and work products.
- Apply the quality concepts, methods, and tools introduced in this chapter to develop a project quality plan.

GLOBAL TECHNOLOGY SOLUTIONS

It was mid-afternoon when Tim Williams walked into the GTS conference room. Two of the Husky Air team members, Sitaraman and Yan, were already seated at the

conference table. Tim took his usual seat, and asked "So how did the demonstration of the user interface go this morning?"

Sitaraman glanced at Yan and then focused his attention on Tim's question. He replied, "Well, I guess we have some good news and some bad news. The good news is that our client was pleased with the work we've completed so far. The bad news, however, is that our prototype did not include several required management reports."

Yan looked at Tim and added, "It was a bit embarrassing because the CEO of the company pointed out our omission. It appears that those reports were specifically requested by him."

Tim looked a bit perplexed and asked, "So how did the client react?"

Sitaraman thought for a moment. "They were really expecting to see those reports," he replied, "but we promised that we would have them ready by next week. The CEO wasn't too happy to hear that it would take another week before we could add the reports and demonstrate the prototype again. However, everyone seemed pleased with what we were able to show them so far, and I think that helped buy us some time."

Tim took out his PDA and studied the calendar for a few minutes. Looking up, he asked, "So how will this impact our schedule?"

Yan opened the folder in front of her and found a copy of the project plan. She answered, "I wondered the same thing myself, and so I took a look at the original baseline project plan. The developers can begin working on what we've finished so far, but it looks like Sitaraman and I will have to work a few late nights this week and probably the weekend. That should get us back on track with minimal impact on the schedule."

Sitaraman sighed and said, "So much for going to the concert this evening. Do you know of anyone who would be interested in two tickets?" That brought a chuckle from the three team members.

Tim smiled and replied, "I'm glad to see that you both handled the situation fairly well and that you thought of a way to keep the project on track, even if it means some overtime for the two of you. However, I think we need to talk about why this problem occurred in the first place and what we can do to reduce the likelihood of similar problems happening again in the future."

Yan gave Tim's words a few seconds to sink in. "After our meeting with the client I talked to a few of the other members of the team," she said. "It turns out that the reports Husky Air's management wanted to see were defined in the requirements document. Unfortunately, several people were working on the same document, and we were given an earlier version of the document that didn't contain the entire specifications for the reports. As a result, we didn't even know the reports were part of the requirements and, therefore, didn't include them in the user interface prototype. I guess we should have checked with the other team members, but we were too busy just trying to get the prototype to work properly."

Tim stood up and walked over to the white board. He then wrote Quality, Verification/Validation, and Change Control on the board. Yan and Sitaraman gave Tim their full attention as he explained, "It seems that having several people work on the same documents, programs, or database files is a common problem. Often two people work on the same document or file at the same time without knowledge of what the other is doing. For example, let's say that person A is working on one section of a document or file, while person B is working on another. If person A saves the document or file to the server and then person B saves her or his document or file to the server afterwards, the changes to Person A's document or file are lost."

"That appears to be exactly what happened to the requirements document we used to develop the prototype!" exclaimed Yan.

"In fact," Sitaraman added, "Yan and I ran into a similar problem when we were working on the prototype. We had several versions of a program that we were developing, but it became confusing as to which version was the latest."

Tim turned to the two team members and said, "As I said before, this seems to be a common problem whenever several team members are working with the same files. What we need is a tool and a method for checking documents out and back in so that we reduce the likelihood of the errors we talked about."

Both Sitaraman and Yan agreed that this was a good idea. Sitaraman then interjected, "Tim, you have 'Verification and Validation' written on the board. Can you expand upon your idea?"

Tim glanced at the board and then turned his attention back to Sitaraman and said, "Sure. We often think of testing as being one of the last activities in software development. But catching problems and errors earlier in the project life cycle are easier and less expensive to fix. Moreover, by the time those problems or errors reach the client, it's too late and can be somewhat embarrassing, as the two of you found out this morning. We need to ask two important questions with respect to each project deliverable, Are we building the right product? And are we building the product the right way? These two questions are the foundation for verification and validation and should be part of an overall quality plan for the project."

Yan thought for a moment and said, "I remember learning about total quality management when I was in school. From what I recall, a lot of this quality stuff really focuses on the customer. But I think we need to rethink our idea of who exactly is our customer."

Both Sitaraman and Tim looked confused. Sitaraman was the first to speak. "But isn't Husky Air *our* customer?"

Yan knew she would have to explain. "Yes, they are, but they are our *end* customer. The team members who carried out the requirements definition and wrote the requirements document didn't realize that you and I were *their* customers because we needed a complete and accurate set of requirements in order to develop the prototype. In turn, the prototype that we develop will be handed off to several other team members who will use it to develop the application system. Subsequently, they will be our customers. I guess we can view the whole project as a customer chain that includes all of the project stakeholders"

"That is a very interesting idea, Yan!" said Tim. "We can build the concepts of quality, verification/validation, and change control into each of the project activities as part of an overall quality plan." The three members of the team felt they had discovered something important that should be documented and shared with the other members of GTS.

Tim replaced the cap on the dry erase pen and said, "It looks like we all have our work cut out for us this next week. While the two of you are busy working on the prototype for your presentation next week, I'll be working late developing a project quality management plan. By the way, do you know of anyone who would be interested in two tickets to a hockey game?"

Things to Think About:

1. What role does quality play in the IT project methodology?
2. How does verification/validation and change control support quality in an IT project?
3. Why should the project team focus on both internal and external customers?

INTRODUCTION

What is quality? Before answering that question, keep in mind that quality can mean different things to different people. For example, if we were comparing the quality of two cars—an expensive luxury car with leather seats and every possible option to a lower-priced economy car that basically gets you where you want to go—many people may be inclined to say that the more expensive car has higher quality. Although the more expensive car has more features, you may not consider it a bargain if you have to keep bringing it back to the shop for expensive repairs. The less-expensive car may start looking much better to you if it were more dependable or met higher safety standards. On the other hand, why do car manufacturers build different models of cars with different price ranges? If everyone could afford luxury cars, then quality comparisons among different manufacturers' cars would be much easier. Although you may have your eyes on a luxury car, your current financial situation (and subsequent logic) may be a constraint. You may have to buy a car you can afford.

Therefore, it is important not to define quality only in terms of features or functionality. Other attributes such as dependability or safety may be just as important to the customer. Similarly in software development, we can build systems that have a great deal of functionality, but perform poorly. On the other hand, we can develop systems that have few features or limited functionality, but also fewer defects.

However, we still need a working definition of quality. The dictionary defines quality as "an inherent or distinguishing characteristic; a property," or as something "having a high degree of excellence." In business, quality has been defined in terms of "fitness for use" and "conformance to requirements." "Fitness for use" concentrates on delivering a system that meets the customer's needs, while "conformance to requirements" centers more on meeting some predefined set of standards. Therefore, quality depends on the needs or expectations of the customer. It is up to the project manager and project team to accurately define those needs or expectations, while allowing the customer to remain within his or her resource constraints.

Although the concepts and philosophies of quality have received a great deal of attention over the last fifty years in the manufacturing and service sectors, many of these same ideas have been integrated into a relatively new discipline or knowledge area called project quality management (PQM). The Project Management Body of Knowledge defines PQM as:

> The processes required to ensure that the project will satisfy the needs for which it was undertaken. It includes all activities of the overall management function that determine the quality policy, objectives, and responsibility and implements them by means of quality planning, quality assurance, quality control, and quality improvement, within the quality system. (95)

Moreover, PMBOK defines the major quality management processes as:

- *Quality Planning*—Determining which quality standards are important to the project and deciding how these standards will be met.
- *Quality Assurance*—Evaluating overall project performance regularly to ensure that the project team is meeting the specified quality standards.
- *Quality Control*—Monitoring the activities and results of the project to ensure that the project complies with the quality standards. In addition, the project organization as a whole should use this information to eliminate causes of unsatisfactory performance and implement new processes and techniques to improve project quality throughout the project organization.

Therefore, PQM should focus on both the *product* and **process** of the project. From our point of view, the project's most important product is the information system solution that the project team must deliver. The system must be "fit for use" and "conform to specified requirements" outlined in both the project's scope and requirements definition. More importantly, the IT product must add measurable value to the sponsoring organization and meet the scope, schedule, and budget objectives. Quality can, however, also be built into the project management and software development processes. A process refers to the activities, methods, materials, and measurements used to produce the product or service. We can also view these processes as part of a quality chain where outputs of one process serve as inputs to other project management processes (Besterfield, Besterfield-Michna et al. 1999).

By focusing on both the product and chain of project processes, the project organization can use its resources more efficiently and effectively, minimize errors, and meet or exceed project stakeholder expectations. The cost of quality, however, can be viewed as the cost of conforming to standards (i.e., building quality into the product and processes) as well as the cost of not conforming to the standards (i.e., rework). Substandard levels of quality can be viewed as waste, errors, or the failure to meet the project sponsor's or client's needs, expectations, or system requirements (Kloppenborg and Petrick 2002).

Failing to meet the quality requirements or standards can have negative consequences for all project stakeholders and impact the other project objectives. More specifically, adding additional work or repeating project activities will probably extend the project schedule and expand the project budget. According to Barry Boehm (Boehm 1981), a software defect that takes one hour to fix when the systems requirements are being defined will end up taking one hundred hours to correct if not discovered until the system is in production. Moreover, poor quality can be an embarrassment for the project manager, the project team, and the project organization. For example, one of the most widely publicized software defect stories was the faulty baggage-handling software at the Denver International Airport. Bugs in the software delayed the opening of the airport from October 1993 to February 1995 at an estimated cost of $1,000,000 a day! Newspaper accounts reported that bags were literally chewed up and contents of bags were flying through the air (Williamson 1997).

The concepts and philosophies of quality management have received a great deal of attention over the years. Although popularized by the Japanese, many organizations in different countries have initiated quality improvement programs. Such programs include ISO certification, six steps to Six Sigma initiatives, or awards such as the Deming Prize or the Malcolm Baldridge National Quality Award. More recently, the Capability Maturity Model (CMM) has provided a framework for software quality that focuses on assessing the process maturity of software development within an organization. Based on writings and teachings of such quality gurus as Shewhart, Deming, Juran, Ishikawa, and Crosby, the core values of these quality programs have a central theme that includes a focus on the customer, incremental or continuous improvement, problem detection and correction, measurement, and the notion that prevention is less expensive than inspection. A commitment to these quality initiatives, however, often requires a substantial cultural change throughout the organization.

In this chapter, you will learn how the concepts of quality management can be applied to IT project management. We will also extend these concepts to include a broader view of PQM in order to support the overall project goal and objectives. As illustrated in Figure 10.1, PQM will not only include the concepts, teachings, tools, and methods of quality management, but also validation/verification and change control.

Verification and validation (V&V) activities within PQM should be carried out throughout the project life cycle. They require the project team to continually ask, Are

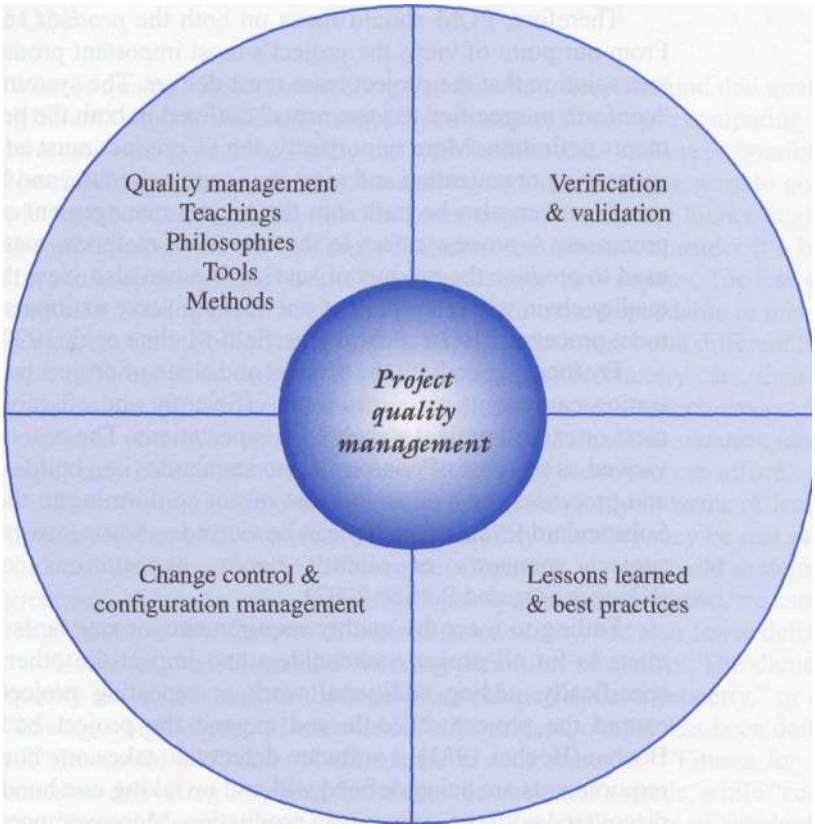


Figure 10.1 Project Quality Management

we building the right product? Are we building the product the right way? Therefore, the project quality plan should not only focus on final testing of the system at the end of the project life cycle, but also on all project deliverables. Finding and fixing problems earlier in the project life cycle is less costly than having to deal with them in the later stages of the project. Finding problems early not only leads to less rework later, but also saves the project manager and project team from having to deal with embarrassing issues and problems once the project's product is in the hands of the project sponsor or end-customer.

In addition, software development often requires a number of people to work on multi-versions of documents, programs, and database files that are shared and distributed among various project stakeholders. Change control in the form of configuration management, therefore, is a method of code and document management to track and organize the different versions of documents and files. It keeps the project team more focused and reduces the likelihood of errors.

In addition, knowledge management and the lessons learned can be implemented as best practices and incorporated in projects throughout the organization. Such changes lead to both continuous improvement and to a maturing of IT project management processes. Taken together, the concepts of quality management, V&V activities, change control, and knowledge management support the overall PQM plan. The plan not only helps improve the overall quality of the project's product and processes, but can also lead to a competitive advantage for the project organization because the project will have a greater likelihood of achieving its expected organizational value and support the scope, schedule, and budget objectives.

QUALITY?

Many experts believe that the quality of software is poor and only getting worse. In fact, the IS manager may give quality low priority because of budget cuts, increasing user demands, competitive pressures, and fast-changing technology. Too often the emphasis is on testing at the end of the development process, while ignoring cost-effective ways to detect and prevent defects earlier in the project. Although testing is important, a true quality practitioner is also interested in process improvement. An emphasis on process improvement is lacking at many organizations that produce poor software. These organizations tend to believe that the right people or right technology will deliver high quality software, but it is technology, people, and process that bring success. The key, therefore, is strong leadership and a commitment to quality throughout the organization. Ignorance and schedule pressure are enemies of quality. Many IS managers have been brainwashed into believing that meeting deadlines is preferable to getting it right the first time. Software defects are acceptable because they can always be fixed later during maintenance. Compounding the problem, IS managers often resist quality improvement programs because they fear failure and see the benefits as intangible. Many vendors' claims *of silver bullet* tools confuse things even more. To overcome these problems, IS managers should get formal training in quality methods and hire one or two trained quality experts. Just having trained quality experts on board, however, does not guarantee success. These people must be given the respect and authority to do their jobs effectively. IS managers should not be intimidated by the effort to develop a comprehensive quality improvement program. They should choose a problem area that will result in the most benefit and be the most likely to result in success. In the end, it is important that the quality concepts be sold to management by showing them how quality improvements provide direct savings to the bottom line.

SOURCE: Adapted from Gary H. Anthes, Quality?! What's That?, *Computerworld,* October 13, 1997. http://www.computerworld.com/news/1997/story/0,11280,9974,00.html

The remainder of this chapter will focus on introducing and delving into several PQM concepts. It includes an overview of the quality movement and a brief history of the people who provided the cornerstones for quality initiatives. It also provides an overview of several quality systems. Finally, it gives a framework to support PQM that integrates the concepts and philosophies of quality, as well as the concepts of software testing, configuration management, and knowledge management.

THE QUALITY MOVEMENT

In this section, we will focus on the concepts associated with quality management, and the history and people who helped shape this important area. This knowledge may help us to better understand how to apply these concepts, ideas, and tools to IT projects.

Craftsmanship

Since the dawn of early humankind, quality was synonymous with craftsmanship. For the earliest Homo sapiens, the quality of the tools and weapons often determined one's survival. Parts could be interchanged to a limited degree, but people generally built things their own way and the products of their labor were highly customized.

This idea was formalized in the Middle Ages when the quality of products and the process to produce those products were held in high esteem. Guilds were created by merchants and artisans for each trade or craft. These unions of the past regulated who could sell goods or practice a trade in a particular town. Members of a guild charged similar prices for products of similar quality and ensured that there were never more craftsmen of a particular trade in a town than could make a decent living. If a worker became ill or too old to work, the guild supported him and his family.

Guilds also ensured the quality of a particular good by regulating the forms of labor. Members of the guild were classified as masters, apprentices, and journeymen. The masters owned the shops and trained the apprentices. An apprentice was bound to a master craftsman, but conditions of control were set by the guild. Training for apprentices required several years, and those who wanted to become master craftsmen had to demonstrate the quality of their work. The number of masters was also limited by the guild. Journeymen were those who had completed their apprenticeship training but were waiting to become masters.

The Industrial Revolution

Eli Whitney (1765-1825) is widely remembered as the inventor of the cotton gin—a machine that could clean the seed from cotton. Whitney's greatest contribution, however, may be the concept of mass producing interchangeable parts. In 1798, Whitney received a contract for $134,000 from the U. S. government to deliver ten thousand rifles within two years. At that time, guns were crafted by gunsmiths, and each gunsmith crafted the pieces differently from other gunsmiths. A lack of gunsmiths and the time required to build a rifle made it impossible to meet the terms of the contract. Time was critical because the United States was anticipating a war with France.

Faced with this problem, Whitney came up with the idea of a new production method in which individual machines could produce each part. Men could then be trained to operate the machines, and the guns could be assembled with parts that met certain tolerance limits. The men operating the machines would, therefore, not be required to have the highly specialized skills of a gunsmith. Whitney called this new production system and division of labor a *manufactory*.

Fortunately, the war between the United States and France never happened. It took Whitney almost a year to develop the manufactory, and then the weather, yellow fever epidemics, delays in obtaining raw materials, and ongoing cotton gin patent lawsuits delayed the implementation of the new production system (Woodall, Rebuck et al. 1997). However, Whitney was able to convince President John Adams of the importance of this innovative approach and subsequently obtained the government's investment and support. Although it took more than ten years to deliver the last rifle, Whitney demonstrated the feasibility of his system and established the seed for the modern assembly line.

Frederic W. Taylor (1856-1915)

As a young man, Frederic W. Taylor worked as an apprentice at the Enterprise Hydraulics Shop. Supposedly, he was told by the older workers how much he should produce each day—no more, no less (Woodall, Rebuck et al. 1997). The workers were paid on a piece rate basis, and if they worked harder or smarter, management would change the production rates and the amount a worker would be paid. These arbitrary rates, or *rules of thumb*, restricted output, and workers produced well below their potential.

Later, as an engineer, Taylor became one of the first to systematically study the relationships between people and tasks. He believed that the production process could become more efficient by increasing the specialization and the division of labor. Using an approach called **scientific management,** Taylor believed that a task could be broken down into smaller tasks and studied to identify the best and most efficient way of doing each subtask. In turn, a supervisor could then teach the worker and ensure that the worker did only those actions essential for completing the tasks, in order to remove human variability or errors. At that time, most workers in U. S. factories were immigrants, and language barriers created communication problems among the workers,

SIXTY-THREE THOUSAND KNOWN BUGS IN WINDOWS 2000?

In February 2000, a Microsoft Corp. memo caused quite a stir when it was leaked to the public. The memo was written by Marc Lucovsky, a Microsoft development manager, and an excerpt from that memo reads:

> Our customers do not want us to sell them products with over sixty-three thousand potential defects. They want those defects corrected. How many of you would spend $500 on a piece of software with over sixty-three thousand potential known defects?

Although it is virtually impossible to produce a piece of software of any size and complexity bug free, Microsoft received its share of bad press, especially as the leak coincided with a proposal in the State of Virginia's General Assembly to pass the Uniform Computer Information Transactions Act (UCITA). The Microsoft memo served as an example of how this act could benefit software vendors to the detriment of the customer. Many consumer and professional organizations opposed this legislation on the grounds that (1) a software vendor could legally disclaim any obligation to sell products that work, (2) in the event of a dispute, a software vendor could disable a customer's software remotely—even if it totally disrupted the customer's business, (3) security experts would be prohibited from reverse engineering software in order to examine it for defects and viruses, and (4) a software vendor could legally stop a user from making public comments on the quality or performance of a product.

Microsoft insisted that the memo was intended to motivate the Windows development team after the source code was scanned using a tool called Prefix. According to Ken White, director of Windows marketing at Microsoft, Prefix flagged code that could be made more efficient in the next release, detected false positives, and analyzed 10 million lines of test code that was not even part of the release. Moreover, White used an analogy of running a grammar-check tool on F. Scott Fitzgerald's classic *The Great Gatsby*—although the tool may highlight unfamiliar words, it doesn't change the content of the novel. With over 750,000 beta testers and security analysts testing Windows 2000, White insisted that the product was "rock solid" and that "the claims are taken out of context and completely inaccurate."

SOURCE: Adapted from Ann Harrison, Microsoft Disputes Reports of 63,000 Bugs in Windows 2000, *Computerworld,* February 16, 2000, http://www.computerworld.eom/news/2000/story/0,11280,43022,00 .html; Frankly Speaking, Win 2K or Win 63K, *Computerworld,* February 21, 2000, http://www.computerworld.com/news/2000/story /0,11280 ,41418.00.html; Ann Harrison and Dominique Deckmyn, Win 2K Bug Memo Causes Brief Uproar, *Computerworld,* February 21, 2000, http://www.computerworld.eom/news/2000/story/0,11280,41419 ,00.html; Dan Gillmor, UCITA Is Going to Hurt You If You Don't Watch Out, *Computerworld,* July 26, 1999, http://www.computerworld.com /news/1999/story/0,11280,36469,00.html.

their supervisors, and even with many coworkers. The use of a stopwatch as a basis for time-motion studies provided a more scientific approach. Workers could produce at their full potential, and arbitrary rates set by management would be removed. To be successful, Taylor also believed that the scientific management approach would require a spirit of cooperation between the workers and management.

Although the scientific management approach became quite popular, it was not without controversy. Many so-called efficiency experts ignored the human factors and tended to believe that profits could be increased by speeding up the workers. Dehumanizing the workers led to conflict between labor and management that eventually laid the foundation for labor unions. Just three years before Taylor died, he acknowledged that the motivation of a person can affect output more than just engineered improvements (Woodall, Rebuck et al. 1997).

Walter A. Shewhart (1891-1967)

In 1918, Walter Shewhart went to work at the Western Electric Company, a manufacturer of telephone equipment for Bell Telephone. At the time, engineers were working to improve the reliability of telephone equipment because it was expensive to repair amplifiers and other equipment after they were buried underground. Shewhart believed that efforts to control production processes were impeded by a lack of information.

Shewhart also believed that statistical theory could be used to help engineers and management control variation of processes. He also reasoned that the use of tolerance limits for judging quality was short-sighted because it provided a method of judging quality for products only after they were produced (Woodall, Rebuck et al. 1997). In 1924, Shewhart introduced the **control chart** as a tool to better understand variation and to allow management to shift its focus away from inspection and more toward the prevention of problems and the improvement of processes.

A control chart provides a picture of how a particular process is behaving over time. All control charts have a center line and control limits on either side of the center line. The center line represents the observed average, while the control limits on either side provide a measure of variability. In general, control limits are set at ±3cr (i.e., ±3 sigma) or ±3s, where a represents the population standard deviation and s represents the sample standard deviation. If a process is normally distributed, control limits based on three standard deviations provides .001 probability limits.

Variation attributed to common causes is considered normal variation and exists as a result of normal interactions among the various components of the process—i.e., chance causes. These components include people, machines, material, environment, and methods. As a result, common cause variation will remain stable and exhibit a consistent pattern over time. This type of variation will be random and vary within predictable bounds.

If chance causes are only present, the probability of an observation falling above the upper control limit would be one out of a thousand, and the probability of an observation falling below the lower control limit would be one out of a thousand as well. Since the probability is so small that an observation would fall outside either of the control limits by chance, we may assume that any observation that does fall outside of the control limits could be attributed to an **assignable cause.** Figure 10.2 provides an example of a control chart where a process is said to be stable or in **statistical control.** Variations attributed to assignable causes can create significant changes in the variation patterns because they are due to phenomenon not considered part of the normal process. An example of assignable cause variation can be seen by the pattern in Figure 10.3. This type of variation can arise because of changes in raw materials, poorly trained people, changes to the work environment, machine failures, inadequate methods, and so forth (Florae, Park et al. 1997). Therefore, if all assignable causes are removed, the process will be stable because only chance factors remain.

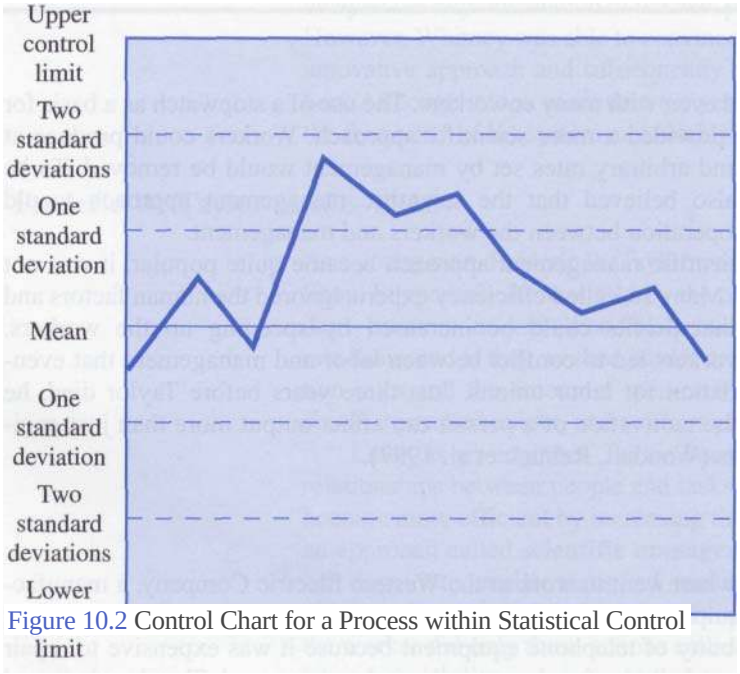


Figure 10.2 Control Chart for a Process within Statistical Control

To detect or test whether a process is not in a state of statistical control, one can examine the control chart for patterns that suggest nonrandom behavior. Florae and his colleagues suggest several tests that are useful for detecting these patterns:

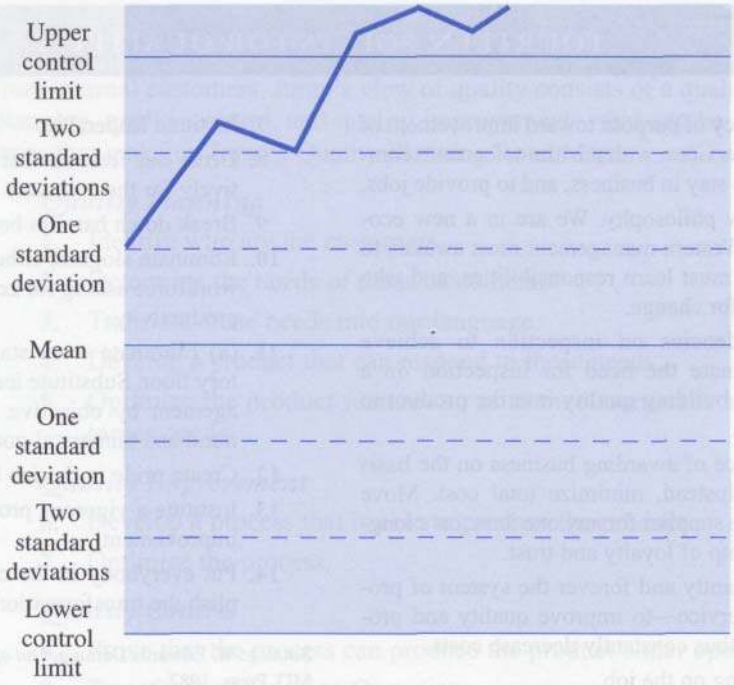


Figure 10.3 Control Chart for a Process Not in Statistical Control

- A single point falls outside the 3σ control limits.
- Although we won't get too bogged down in statistics, we can look for patterns that suggest that the observed data are not statistically independent. A process may not be in control if we observe:
  - At least two of three successive values that fall on the same side of and more than two standard deviations from the centerline.
  - At least four out of five successive values that fall on the same side of and more than one standard deviation away from the centerline.
  - At least eight successive values that fall on the same side of the centerline.

Control charts are a valuable tool for monitoring quality; however, it is important to keep in mind that one can see patterns where patterns may not exist (Florac, Park et al. 1997).

W. Edwards Deming (1900-1993)

While working at the Western Electric Hawthorne plant in Chicago, Illinois, during the 1920s, Deming became aware of the extensive division of labor. Management tended to treat the workers as just another cog in the machinery. Moreover, the workers were not directly responsible for the quality of the products they produced. Final inspection was used as a means to control quality and reductions in the per piece rate reflected scrap and rework.

Deming met Shewhart while working at Bell Laboratories in New Jersey in the 1930s and became interested in Shewhart's application of statistical theory. Deming realized that costly inspections could be eliminated if workers were properly trained and empowered to monitor and control the quality of the items they produced.

FOURTEEN POINTS FOR QUALITY

1. Create constancy of purpose toward improvement of products and services, with the aim to become com petitive, and to stay in business, and to provide jobs.
2. Adopt the new philosophy. We are in a new eco nomic arena. Western management must awaken to the challenge, must learn responsibilities, and take on leadership for change.
3. Cease dependencies on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place.
4. End the practice of awarding business on the basis of price tag. Instead, minimize total cost. Move toward a single supplier for any one item, on a long-term relationship of loyalty and trust.
5. Improve constantly and forever the system of pro duction and service—to improve quality and pro ductivity, and thus constantly decrease costs.
6. Institute training on the job.
7. Institute leadership.
8. Drive out fear, so that everyone may work effec tively for the company.
9. Break down barriers between departments.
10. Eliminate slogans, exhortations, and targets for the workforce asking for zero defects and new levels of productivity.
11. (a) Eliminate work standards (quotas) on the fac tory floor. Substitute leadership, (b) Eliminate man agement by objective. Eliminate management by numbers, numerical goals. Substitute leadership.
12. Create pride in the job being done.
13. Institute a vigorous program of education and self-improvement.
14. Put everybody in the company to work to accom plish the transformation.

SOURCE: W. Edwards Deming, *Out of the Crisis*, Cambridge, MA: The MIT Press, 1982.

Deming and his teachings were relatively unnoticed in the United States. Soon after World War II, Japan was a country faced with the challenge of rebuilding itself after devastation and military defeat. Moreover, Japan had few natural resources so the export of manufactured goods was essential. Unfortunately, the goods that it produced were considered inferior in many world markets.

To help Japan rebuild, a group called the Union of Japanese Scientists and Engineers (JUSE) was formed to work with U. S. and allied experts to improve the quality of the products Japan produced. As part of this effort, in the 1950s Deming was invited to provide a series of day-long lectures to Japanese managers. The focus of these lectures was statistical control and quality. The Japanese embraced these principles, and the quality movement acquired a strong foothold in Japan. In tribute to Deming, the Japanese even named their most prestigious quality award the Deming Prize.

Until the 1970s, Deming was virtually unknown in the West. In 1980, an NBC documentary entitled "If Japan Can, Why Can't We" introduced him and his ideas to his own country and the rest of the world. Many of Deming's philosophies and teachings are summarized in his famous fourteen points for quality that are outlined and discussed in his book *Out of the Crisis* (Deming 1982).

Joseph Juran (1904-)

Joseph Juran's philosophies and teachings have also had an important and significant impact on many organizations worldwide. Like Deming, Juran started out as an engineer in the 1920s. In 1951 he published the *Quality Control Handbook*, which viewed quality as "fitness for use" as perceived by the customer. Like Deming, Juran was invited to Japan by JUSE in the early 1950s to conduct seminars and lectures on quality.

Juran's message on quality focuses on his belief that quality does not happen by accident—it must be planned. In addition, Juran distinguishes external customers from internal customers. Juran's view of quality consists of a quality trilogy—quality planning, quality control, and quality improvement—that can be combined with the steps that make up Juran's Quality Planning Road Map.

Quality Planning

1. Identify who are the customers.
2. Determine the needs of those customers.
3. Translate those needs into our language.
4. Develop a product that can respond to those needs.
5. Optimize the product features so as to meet our needs as well as cus tomer needs.

Quality Improvement

6. Develop a process that is able to produce the product.
7. Optimize the process.

Quality Control

8. Prove that the process can produce the product under operating conditions.
9. Transfer the process to Operations.

Kaoru Ishikawa (1915-)

Kaoru Ishikawa studied under Deming and believes that quality improvement is a continuous process that depends heavily on all levels of the organization—from top management down to every worker performing the work. In Japan this belief led to the use of quality circles that engaged all members of the organization. In addition to the use of statistical methods for quality control, Ishikawa advocated the use of easy-to-use analytical tools that included cause-and-effect diagrams (called the Ishikawa diagram, or fishbone diagram, because it resembles the skeleton of a fish), the Pareto Chart, and flow charts.

Although the Ishikawa, or fishbone, diagram was introduced in an earlier chapter, it can be used in a variety of situations to help understand various relationships between causes and effects. An example of an Ishikawa diagram is illustrated in Figure 10.4. The effect is the rightmost box and represents the problem or characteristic that requires improvement. A project team could begin by identifying the major causes, such as people, materials, management, equipment, measurements, and environment, that may influence the problem or quality characteristic in question. Each major cause can then be subdivided in potential sub-causes. For example, causes associated with people may be lack of training or responsibility in identifying and correcting a particular problem. An Ishikawa diagram can be best developed by brain-storming or by using a learning cycle approach. Once the diagram is complete, the project team can investigate the possible causes and recommend solutions to correct the problems and improve the process.

Another useful tool is a **Pareto diagram,** which was developed by Alfred Pareto (1848-1923). Pareto studied the distribution of wealth in Europe and found that about 80 percent of the wealth was owned by 20 percent of the population. This idea has held in many different settings and has become known as the 80/20 rule. For example, 80 percent of the problems can be attributed to 20 percent of the causes.

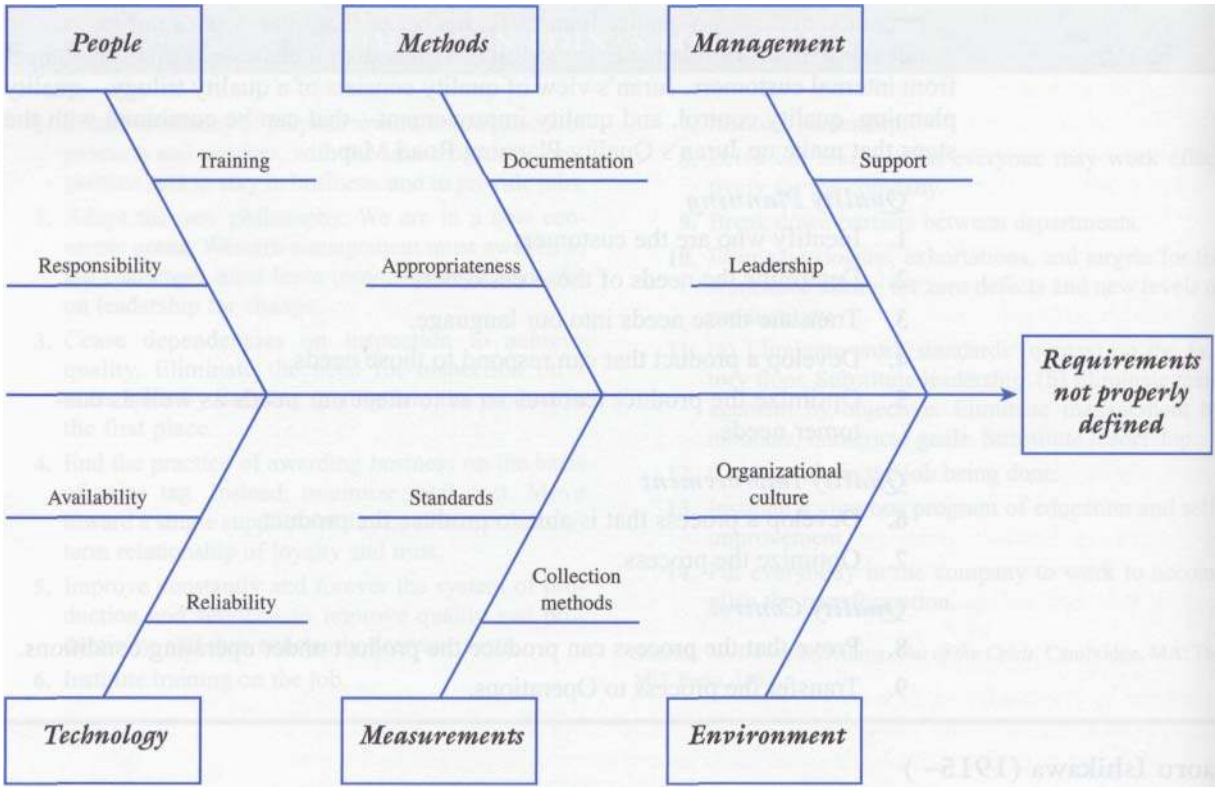


Figure 10.4 Ishikawa, or Fishbone, Diagram

Pareto diagrams can be constructed by (Besterfield, Besterfield-Michna et al. 1999):

1. Determining how the data will be classified. It can be done by the nature of the problem, the cause, non-conformity, or defect or bug.
2. Determining whether frequency, dollar amount, or both should be used to rank the classifications.
3. Collecting the data for an appropriate time period.
4. Summarizing the data by rank order of the classifications from largest to smallest, from left to right.

Pareto diagrams are useful for identifying and investigating the most important problems by ranking problems in descending order from left to right. For example, let's say that we have tracked all the calls to a call center over a period of one week. If we were to classify the different types of problems and graph the frequency of each type of call, we would end up with a chart similar to Figure 10.5.

As you can see, the most frequent type of problem had to do with documentation questions. In terms of quality improvement, it may suggest that the user documentation needs to be updated.

In addition, **flow charts** can be useful for documenting a specific process in order to understand how products or services move through various functions or operations. A flow chart can help visualize a particular process and identify potential problems or bottlenecks. Standardized symbols can be used, but are not necessary. It is more important to be able to identify problems or bottlenecks, reduce complexity, and determine who is the next customer (Besterfield, Besterfield-Michna et al. 1999).

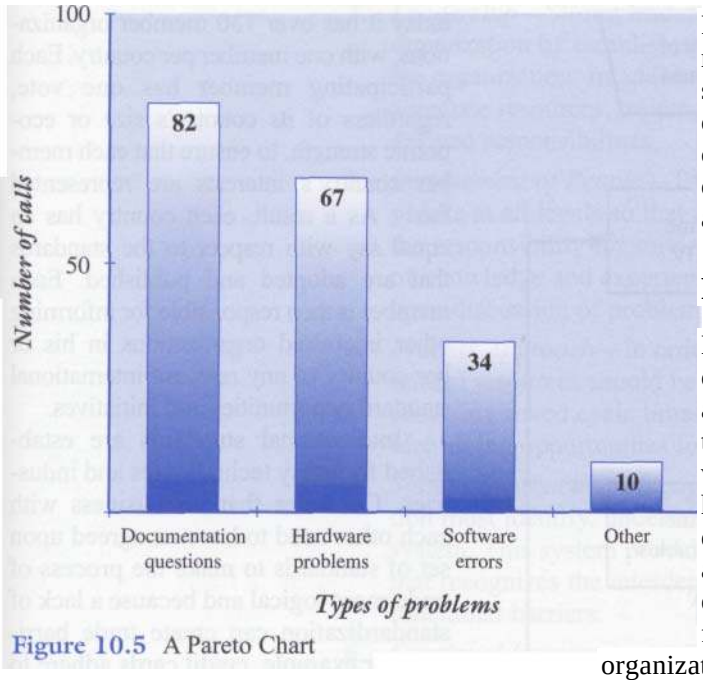


Figure 10.5 A Pareto Chart

Figure 10.6, for example, documents the project management process for verifying a project's scope. The original customer who initiates the original project request might be the project's client or sponsor. The customer who receives the output of the scope verification process might be a specific member of the project team.

Phillip Crosby (1926-2001)

Like F.W. Taylor, Philip Crosby developed many of his ideas from his experiences working on an assembly line. After serving in the Navy during the Korean War, he worked his way up in a variety of quality control positions until he held the position of corporate vice president and director of quality for ITT. In 1979, he published a best-selling book, *Quality is Free*, and eventually left ITT to start his own consulting firm that focused on teaching other organizations how to manage quality.

Crosby defined quality as conformance to requirements based on the customer's needs and advocated a top-down approach to quality in which it is management's responsibility to set a quality example for workers to follow. Crosby also advocated "doing it right the first time" and "zero defects", which translate into the notions that quality is free and that non-conformance costs organizations money.

| QUALITY SYSTEMS

Although guilds were the first organizations to ensure quality standards, there are a number of different organizations and approaches for defining and implementing quality standards in organizations. **Standards** are documented agreements, protocols, or rules that outline the technical specifications or criteria to be used to ensure that products, services, processes, and materials meet their intended purpose. Standards also provide a basis for measurement because they provide criterion, or basis, for comparison.

International Organization for Standardization (ISO)

One of the most widely known standards organizations is the International Organizations for Standardization (ISO). Although you may think the acronym should be IOS, the name for the organization is ISO and was derived from the Greek word *isos*, which means *equal*. The name avoids having different acronyms that would result from International Organization for Standardization being translated in different languages.

ISO was officially formed in 1947 after delegates from twenty-five countries met in London the previous year with the intention of creating an international organization whose mission would be "to facilitate the international coordination and unification of industrial standards." ISO is not owned or managed by any national government, and

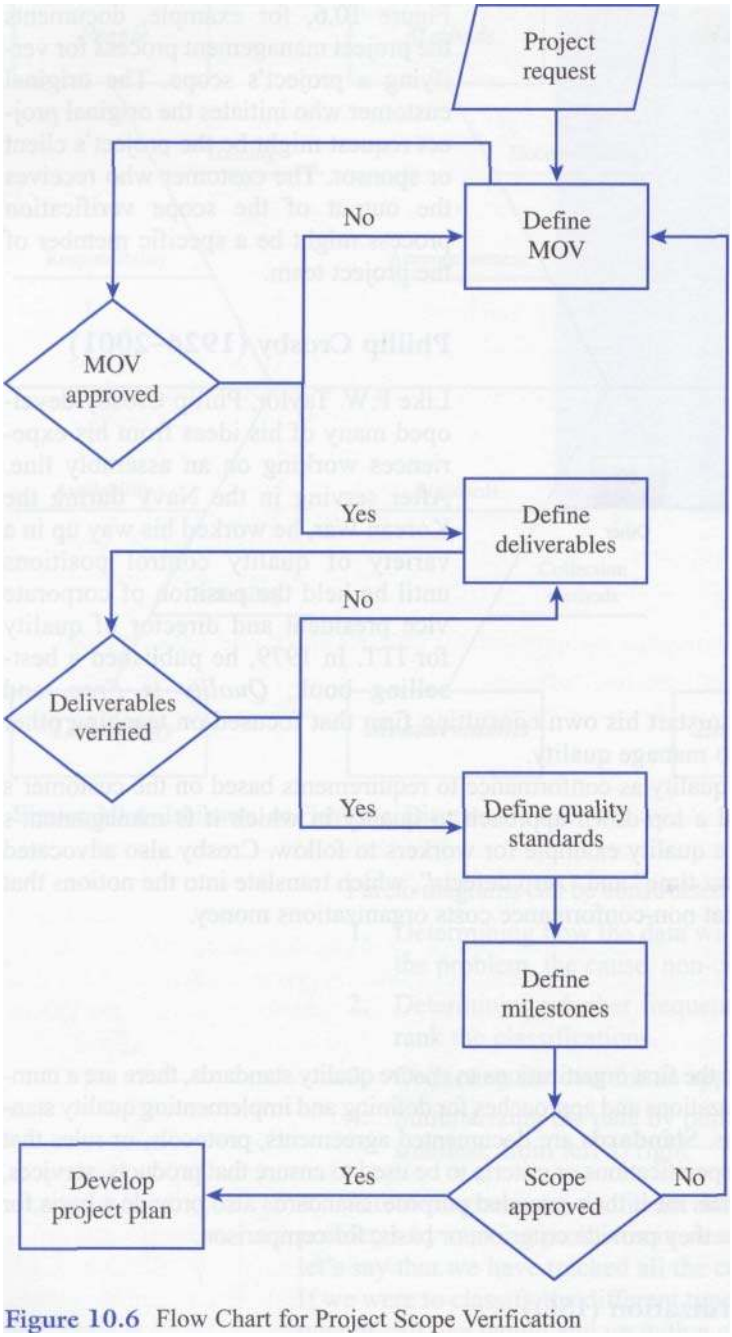


Figure 10.6 Flow Chart for Project Scope Verification

today it has over 130 member organizations, with one member per country. Each participating member has one vote, regardless of its country's size or economic strength, to ensure that each member country's interests are represented fairly. As a result, each country has an equal say with respect to the standards that are adopted and published. Each member is then responsible for informing other interested organizations in his or her country of any relevant international standard opportunities and initiatives.

International standards are established for many technologies and industries. Countries that do business with each other need to have an agreed upon set of standards to make the process of trade more logical and because a lack of standardization can create trade barriers. For example, credit cards adhere to a standard size and thickness so that they can be used worldwide.

Although most of the ISO standards are specific to a particular product, material, or process, a set of standards make up the ISO 9000 and ISO 14000 families. These are known as "generic management system standards" in which the same standards can be applied to any size or type of organization in any industry. The term **management system** refers to the processes and activities that the organization performs. ISO 9000 was originally initiated in 1987 and focuses on quality management with respect to improved customer satisfaction and the continuous improvement of an organization's performance and processes. On the other hand, standards that fall under the ISO 14000 came about in 1997 and are concerned primarily with environmental management—that is, how an organization can minimize any harmful effects on the environment that may be caused by its activities and operations.

The ISO 9000 standards were revised in 2000 (and are now called ISO 9000:2000) and focus on eight quality management principles that provide a framework for organizations:

1. *Customer Focus*—The customer is key for all organizations. Therefore, organizations should strive to meet and exceed the current and future needs of their customers.

2. *Leadership*—Strong leaders create a sense of purpose and direction for an organization by establishing and communicating a vision and mission for the organization. In addition, leaders inspire and provide their people with adequate resources, training, and empowerment to act within a set of well-defined responsibilities.
3. *Involvement of People*—To be successful, an organization must involve people at all levels so that individuals accept ownership for problems and the responsibility for solving them. This involvement requires the sharing of knowledge and experiences freely, while supporting and encouraging the open discussion of problems and issues.
4. *Process Approach*—In order to achieve a desired result, activities and related resources should be managed as a process, which allows for lower costs, improved cycle times, predictable results, and a focused approach for identifying opportunities for improvement.
5. *System Approach to Management*—To achieve its objectives, an organiza tion must identify, understand, and manage its interrelated processes as a system. This system provides a more structured and integrated approach that recognizes the interdependencies among processes and reduces cross-functional barriers.
6. *Continual Improvement*—Continuous improvement of the organization's products, processes, and systems should be a permanent objective. It should entail an organizationwide approach with established goals to guide and measure progress.
7. *Factual Approach to Decision Making*—Decision making should be based on data and facts. Data and information should be accurate and reliable, and should be analyzed using valid methods. However, informed decision making should be balanced between analysis based on facts or data and experience and intuition.
8. *Mutually beneficial supplier relationships*—An interdependent relation ship exists between an organization and its suppliers. This relationship can be mutually beneficial if it increases the ability to create value for both parties. This value can support a long-term relationship that allows for pooling expertise and resources, while improving flexibility and speed in jointly responding to changing markets or customer needs. This relationship requires trust, open communication, and the sharing of information that will support joint activities between an organization and its suppliers.

To show that a product, service, or system meets the relevant standards, an organization may receive a certificate as proof. For example, many organizations have been issued ISO 9000 certificates as testaments that they have quality management systems in place and that their processes conform to the ISO 9000 standards. Keep in mind that these standards focus on processes not products. An organization can be certified in one of three quality systems under ISO 9000:

- *ISO 9001*—For organizations whose business processes range from design through development, as well as production, installation, and service. ISO 9001 contains twenty standards, or requirements, that must be met for a quality system to be in compliance. Although ISO 9001 can be applied to all engineering disciplines, it is the one most relevant to software development.

- *ISO 9002*—For organizations that do not design and develop products. With the exception of design control requirements, the requirements are similar to ISO 9001.
- *ISO 9003*—For organizations whose business processes do not include design control, process control, purchasing, or service. The focus is on inspection and testing of final products and services in order to meet speci fied requirements.

If an organization decides that it would like to be ISO certified as meeting the ISO standards, it usually begins by studying the ISO guidelines and requirements. The organization then conducts an internal audit to make sure that every ISO requirement is met. After deficiencies or gaps are identified and corrected, the organization then has a third party called a **registrar** audit its quality management system. If the registrar finds that the organization meets the specified ISO standards and requirements, it will issue a certificate as a testament that the organization's products and services are managed and controlled by a quality management system that meets the requirements of ISO 9000. ISO does not conduct the audits or issue certificates. In addition, an organization does not have to have a formal registration or certificate to be in compliance with the ISO standards; however, customers may be more likely to believe that an organization has a quality system if an independent third party attests to it.

TickIT

The TickIT initiative began in 1991 following a report on software quality published by the British Department of Trade and Industry. The report reviewed the state of software quality and suggested that many software organizations were reluctant to adopt the ISO 9000 standards because they believed them to be too general or difficult to interpret.

The British government then asked the British Computer Society (BCS) to take on a project called TickIT, which would provide a method for registering software development systems under the ISO 9000 standards. TickIT guides a company through certification of software quality under the ISO 9001 framework. This certification is applicable to all types of information systems that include software development processes—from software houses that produce software as an end product or service to in-house software development supported by an internal IS function.

TickIT certification relates directly to ISO 9001:2000. More than 1,400 ISO 9001/TickIT certificates have been issued worldwide by twelve certification bodies accredited in Britain and Sweden. Certification is conducted by an independent external auditor who has been specially trained under the International Register of Certified Auditors (IRCO), which is supported by the British Computer Society. After being successfully audited by a TickIT certified auditor, an organization receives a certificate that it is in compliance with ISO 9001:2000 and it is endorsed with a TickIT logo. Subsequently, TickIT gives software developers an accredited quality certification specialized to software organizations and, hopefully, increases the confidence of customers and suppliers.

Six Sigma (6σ)

The term *Six Sigma* was originated by Motorola (Schaumburg, Illinois) in the mid-1980s. The concept of Six Sigma came about as a result of competitive pressures by foreign firms that were able to produce higher quality products at a lower cost than Motorola. Even Motorola's own management at that time admitted that "our quality stinks" (Pyzdek 1999).

Sigma (a) is a Greek letter and in statistics represents the standard deviation to measure variability from the mean or average. In organizations, variation is often the cause of defects or out-of-control processes and translates into products or services that do not meet customer needs or expectations. If a manufacturing process follows a normal distribution, then the mean or average and the standard deviation can be used to provide probabilities for how the process can or should perform.

Six Sigma focuses on defects per opportunities (DPO) as a basis for measuring the quality of a process rather than products it produces, because products may vary in complexity. A defect may be thought of as anything that results in customer dissatisfaction. The sigma value, therefore, tells us how often defects are likely to occur. The higher the value of sigma, the lower the probability of a defect occurring. As illustrated in Table 10.1, a value of six sigma indicates that there will only be 3.4 defects per million, while three sigma quality translates to 66,807 defects per million. Table 10.2 provides several real-world examples that compare the differences between three sigma and six sigma quality.

Therefore, Six Sigma can be viewed as a quality objective whereby customer satisfaction will increase as a result of reducing defects; however, it is also a business-driven approach for improving processes, reducing costs, and increasing profits. The key steps in the Six Sigma improvement framework are the **D-M-A-I-C** cycle:

- *Define*—The first step is to define customer satisfaction goals and subgoals—for example, reduce cycle time, costs, or defects. These goals then provide a baseline or benchmark for the process improvement.
- *Measure*—The Six Sigma team is responsible for identifying a set of rele vant metrics.
- *Analyze*—With data in hand, the team can analyze the data for trends, pat terns, or relationships. Statistical analysis allows for testing hypotheses, modeling, or conducting experiments.
- *Improve*—Based on solid evidence, improvements can be proposed and implemented. The *Measure - Analyze - Improve* steps are generally itera tive to achieve target levels of performance.
- *Control*—Once target levels of performance are achieved, control methods and tools are put into place in order to maintain performance.

To carry out a Six Sigma program in an organization, a significant investment in training and infrastructure may be required. Motorola adopted the following martial arts terminology to describe these various roles and responsibilities (Pyzdek 1999):

Table 10.1 Sigma and Defects per Million

Sigma	*Defects Per Million*
1 σ	690,000
2 σ	308,537
3 σ	66,807
4 σ	6,210
5 σ	233
6 σ	3.4

Table 10.2 Comparison of Three Sigma and Six Sigma

3 σ	*6 σ*
Five short or long landings at any major airport	One short or long landing in 10 years at all airports in the U.S.
Approximately 1,350 poorly performed surgical operations in one week	One incorrect surgical operation in 20 years
Over 40,500 newborn babies dropped by doctors or nurses each year	Three newborn babies dropped by doctors or nurses in 100 years
Drinking water unsafe to drink for about 2 hours each month	Water unsafe to drink for one second every six years

- *Master Black Belts*—Master black belts are people within the organization who have the highest level of technical and organizational experience and expertise. Master black belts train black belts and, therefore, must know everything a black belt knows. Subsequently, a maser black belt must have technical competence, a solid foundation in statistical methods and tools, and the ability to teach and communicate.
- *Black Belts*—Although black belts may come from various disciplines, they should be technically competent and held in high esteem by their peers. Black belts are actively involved in the Six Sigma change process.
- *Green Belts*—Green belts are Six Sigma team leaders or project managers. Black belts generally help green belts choose their projects, attend training with them, and then assist them with their projects once the project begins.
- *Champions*—Many organizations have added the role of a Six Sigma cham pion. Champions are leaders who are committed to the success of the Six Sigma project and can ensure that barriers to the Six Sigma project are removed. Therefore, a champion is usually a high-level manager who can remove obstacles that may involve funding, support, bureaucracy, or other issues that black belts are unable to solve on their own

Although the concept of Six Sigma was initially used in a manufacturing environment, many of the techniques can be applied directly to software projects (Siviy 2001). The usefulness of Six Sigma lies in the conscious and methodical way of achieving customer satisfaction through the improvement of current processes and products and their design.

The Capability Maturity Model (CMM)

In 1986, the Software Engineering Institute (SEI), a federally funded research development center at Carnegie Mellon University, set out to help organizations improve their software development processes. With the help of the Mitre Corporation and Watts Humphrey, a framework was developed to assess and evaluate the capability of software processes and their maturity, and the work of the SEI evolved into the Capability Maturity Model (CMM) (Humphrey 1988). The CMM provides a set of recommended practices for a set of key process areas specific to software development. The objective of the CMM is to provide guidance as to how an organization can best control its processes for developing and maintaining software. In addition, the CMM provides a path for helping organizations evolve their current software processes toward software engineering and management excellence (Paulk, Curtis et al. 1993).

To understand how the CMM may support an organization, several concepts must first be defined:

- *Software Process*—A set of activities, methods, or practices and transfor mations used by people to develop and maintain software and the deliverables associated with software projects. Included are such things as project plans, design documents, code, test cases, user manuals, and so forth.
- *Software Process Capability*—The *expected* results that can be achieved by following a particular software process. More specifically, the capability of an organization's software processes provides a way of predicting the out comes that can be expected if the same software processes are used from one software project to the next.
- *Software Process Performance*—The *actual* results that are achieved by following a particular software process. Therefore, the actual results

achieved through software process performance can be compared to the expected results achieved through software process capability.

- *Software Process Maturity*—The extent to which a particular software process is explicitly and consistently defined, managed, measured, controlled, and effectively used throughout the organization.

One of the keys to the CMM is using the idea of software process maturity to describe the difference between immature and mature software organizations. In an immature software organization, software processes are improvised or developed ad hoc. For example, a software project team may be faced with the task of defining user requirements. When it comes time to complete this task, the various members of the team may have different ideas concerning how to accomplish it. Several of the members may approach the task differently and, subsequently achieve different results. Even if a well-defined process that specifies the steps, tools, resources, and deliverables required is in place, the team may not follow the specified process very closely or at all.

The immature software organization is characterized as being reactive; the project manager and project team spend a great deal of their time reacting to crises or find themselves in a perpetual state *of fire fighting.* Schedules and budgets are usually exceeded. As a result, the quality and functionality of the software system and the associated project deliverables are often compromised. Project success is determined largely by who is (or who is not) part of the project team. In addition, immature software organizations generally do not have a way of judging or predicting quality. Since these organizations operate in a perpetual crisis mode, there never seems to be enough time to address problem issues or improve the current processes.

Mature software organizations, on the other hand, provide a stark contrast to the immature software organization. More specifically, software processes and the roles of individuals are defined explicitly and communicated throughout the organization. The software processes are consistent throughout the organization and continually improved based on experimentation or experiences. The quality of each software process is monitored so that the products and processes are predictable across different projects. Budgets and schedules are based on past projects so they are more realistic and the project goals and objectives are more likely to be achieved. Mature software organizations are proactive and they are able to follow a set of disciplined processes throughout the software project.

The CMM defines five levels of process maturity, each requiring many small steps as a path of incremental and continuous process improvement. These stages are based on many of the quality concepts and philosophies of Shewhart, Deming, Juran, and Crosby (Paulk, Curtis et al. 1993). Figure 10.7 illustrates the CMM framework for software process maturity. These levels allow an organization to assess its current level of software process maturity and then help it prioritize the improvement efforts it needs to reach the next higher level (Caputo 1998).

Maturity levels provide a well-defined, evolutionary path for achieving a mature software process organization. With the exception of Level 1, each maturity level encompasses several key process areas that an organization must have in place in order to achieve a particular level of maturity. There are five levels of software process maturity.

Level 1: Initial The initial level generally provides a starting point for many software organizations. This level is characterized by an immature software organization in which the software process is ad hoc and often reactive to crises. Few, if any, processes for developing and maintaining software are defined. The Level 1 software organization does not have a stable environment for software projects, and success of a project rests

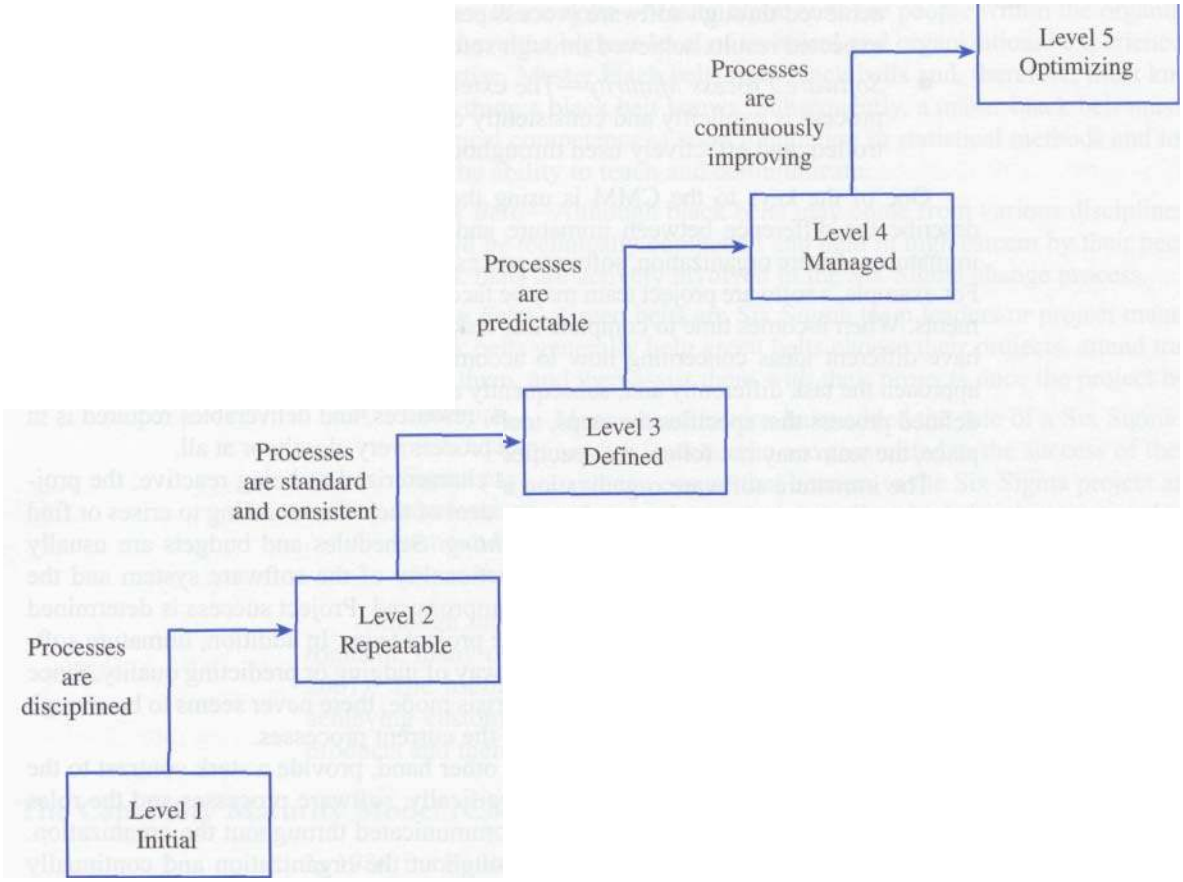


Figure 10.7 Levels of Software Process Maturity

largely with the people on the project and not the processes that they follow. As a result, success is difficult to repeat across different projects throughout the organization.

Key Process Areas

- No key process areas are in place.

Level 2: Repeatable At this level, basic policies, processes, and controls for managing a software project are in place. Project schedules and budgets are more realistic because planning and managing new projects is based upon past experiences with similar projects. Although software processes between projects may be different at this level, the process capability of Level 2 organizations is more disciplined because software processes are more documented, enforced, and improving. As a result, many previous project successes can be repeated by other project teams on other projects.

Key Process Areas

- Software Configuration Management—Supports the controlling and man aging of changes to the various project deliverables and software products throughout the project and software life cycles.
- Software Quality Assurance—Provides project stakeholders with an understanding of the processes and standards used to support the project quality plan.

- Software Subcontract Management—Supports the selection and management of qualified software subcontractors.
- Software Project Tracking and Oversight—Ensures that adequate controls are in place to oversee and manage the software project so that effective decisions can be made and actions taken when the project's actual performance deviates from the project plan.
- Software Project Planning—Establishes realistic plans for software development and managing the project.
- Requirements Management—Ensures that a common understanding of the user's requirements is established and becomes an agreement and basis for planning.

Level 3: Defined At Level 3, software engineering and management processes are documented and standardized throughout the organization and become the organization's standard process. And, a group is established to oversee the organization's software processes and an organizationwide training program to support the standard process is implemented. Thus, activities, roles, and responsibilities are well defined and understood throughout the organization. The software process capability of this level is characterized as being standard, consistent, stable, and repeatable. However, this standard software process may be tailored to suit the individual characteristics or needs of an individual project.

Key Process Areas

- Peer Reviews—Promotes the prevention and removal of software defects as early as possible and is implemented through code inspections, structured walkthroughs, and so forth.
- Intergroup Coordination—Allows for an interdisciplinary approach where the software engineering group participates actively with other project groups in order to produce a more effective and efficient software product.
- Software Product Engineering—Defines a consistent and effective set of integrated software engineering activities and processes in order to produce a software product that meets the users' requirements.
- Integrated Software Management—Supports the integration of software engineering and management activities into a set of well-defined and understood software processes that are tailored to the organization.
- Training Programs—Facilitates the development of individuals' skills and knowledge so that they may perform their roles and duties more effectively and efficiently.
- Organization Process Definition—Supports the identification and development of a usable set of software processes that improve the capability of the organization across all software projects.
- Organization Process Focus—Establishes organizational responsibility for implementing software processes that improve the organization's overall software process capability.

Level 4: Managed At this level, quantitative metrics for measuring and assessing productivity and quality are established for both software products and processes. This information is collected and stored in an organizationwide repository that can be used to analyze and evaluate software processes and products. Control over projects is achieved by reducing the variability of project performance so that it falls within

acceptable control boundaries. The software processes of software organizations at this level are characterized as being quantifiable and predictable because quantitative controls are in place to determine whether the process performs within operational limits. Moreover, these controls allow for predicting trends and identifying when assignable causes occur that require immediate attention.

*Key **Process** Areas*

- Software Quality Management—Establishes a set of processes to support the project's quality objectives and project quality management activities.
- Quantitative Process Management—Provides a set of quantitative or statis tical control processes to manage and control the performance of the soft ware project by identifying assignable cause variation.

*Level **5: Optimizing*** At the highest level of software process maturity, the whole organization is focused on continuous process improvement. These improvements come about as a result of innovations using new technology and methods and incremental process improvement. Moreover, the organization has the ability to identify its areas of strengths and weaknesses. Innovations and best practices based on lessons learned are identified and disseminated throughout the organization.

***Key Process** Areas*

- Process Change Management—Supports the continual and incremental improvement of the software processes used by the organization in order to improve quality, increase productivity, and decrease the cycle time of soft ware development.
- Technology Change Management—Supports the identification of new tech nologies (i.e., processes, methods, tools, best practices) that would be bene ficial to the organization and ensures that they are integrated effectively and efficiently throughout the organization.
- Defect Prevention—Supports a proactive approach to identifying and pre venting software defects.

As an organization's software process maturity increases, the difference between expected results and actual results narrows. In addition, performance can be expected to improve when maturity levels increase because costs and development time will decrease, while quality and productivity increase.

According to the SEI, skipping maturity levels is counter-productive. If an organization was evaluated at Level 1, for example, and wanted to skip to Level 3 or Level 4, it may be difficult because the CMM identifies levels through which an organization must evolve in order to establish a culture and experiences.

Both the CMM and ISO 9001 series of standards focus on quality and process improvement. A technical paper by Mark C. Paulk (1994) compares the similarities and differences between the CMM and ISO 9001. His analysis indicates an ISO 9001-compliant organization would satisfy most of the Level 2 and Level 3 goals. Although Level 1 organizations could be ISO 9001 compliant, it may be difficult for these organizations to remain compliant. In turn, there are many practices in the CMM that are not addressed by ISO 9001, and it is, therefore, possible for a Level 1 organization to be ISO 9001 compliant. A Level 2 organization should have little difficulty in receiving ISO 9001 certification.

After reading this section, you may be wondering which quality system is best. Should an organization focus on ISO certification? Or, should it concentrate its efforts on the CMM? Although the market may dictate a particular certification, an

THE COST OF NOT FOLLOWING DIRECTIONS

Most IT groups have formal guidelines for developing software. Unfortunately, these guidelines are in a thick binder that ends up collecting dust or hidden in someone's desk drawer. According to Software Productivity Research (SPR), a regular inspection of the application design and code can reduce software defects by 50 percent. The difficult part, however, is getting the development team to follow step-by-step instructions for reviews, inspections, or meetings with users. According to Roger Pressman, a software consultant from Orange, Connecticut, many developers view a process as an extraneous activity that one must endure before getting to the cooler part of development using a hot, new technology. But according to Pressman, "the problem is that without a process, you get screwed up just writing code." The problem becomes how to get developers to stick to the processes. One answer is to have them help write it—because people are more likely to follow the process if they are part of developing it. Therefore, the project team should be invited to add to the process any time they come up with a proven, effective technique. The goal of developing a process is not to create binders filled with paper that no one ever looks at, but to deliver projects or software on time, within budget, and that meet or exceed expectations. Although SPR has estimated that a company can save $17 in maintenance costs for every $1 invested up-front on requirements reviews, design and code inspections, and other development processes, the problem is that most developers are rewarded for getting the project done and not for following a process. As a result, developers get the project done any way they can get it done fast.

SOURCE: Adapted from Julia King, Ignoring Development Guidelines Raises Costs, *Computerworld*, MaylS, 1998, http://www.computer-world.com/news/1998/story/0,11280,30906,00.html.

organization should be focused on continuous improvement that leads to competitive advantage and not necessarily on a certificate or maturity level (Paulk 1994).

THE IT PROJECT QUALITY PLAN

All project stakeholders want quality; unfortunately, there is no commonly accepted approach for PQM so many project managers approach it differently (Lewis 2000). Therefore, a basic framework will be introduced here to guide and integrate the knowledge areas of quality planning, quality assurance, quality control, and quality improvement. This framework provides a basic foundation for developing an IT project quality plan to support the project's quality objectives. This plan may be formal or informal, depending on the size of the project; however, the underlying philosophies, standards, and methods for defining and achieving quality should be well-understood and communicated to all project stakeholders. Moreover, the project quality plan should support the project organization, regardless of whether it is attempting to meet ISO or CMM requirements or self-imposed quality initiatives and objectives.

PQM also becomes a strategy for risk management. The objectives of PQM are achieved through a quality plan that outlines the goals, methods, standards, reviews, and documentation to ensure that all steps have been taken to ensure customer satisfaction by assuring them that a quality approach has been taken (Lewis 2000). Figure 10.8 provides a representation of the IT project quality plan discussed in this section.

Quality Philosophies and Principles

Before setting out to develop an IT project quality plan, the project and project organization should define the direction and overall purpose for developing the project quality plan. This purpose should be grounded upon the quality philosophies, teachings, and principles that have evolved over the years. Although several different quality gurus and their

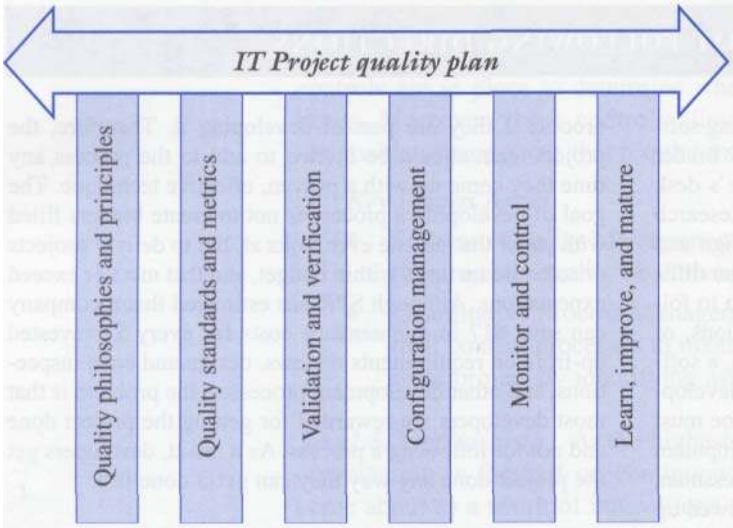


Figure 10.8 The IT Project Quality Plan

teachings were introduced in this chapter, several common themes can provide the backbone for any organization's plan for ensuring quality of the project's processes and product. These ideas include: a focus on customer satisfaction, prevention of mistakes, improving the process to improve the product, making quality everyone's responsibility, and fact-based management.

Focus on Customer Satisfaction Customer satisfaction is the foundation of quality philosophies and concepts. Customers have expectations and are the best judge of quality. Meeting or exceeding those expectations can lead to improved customer satisfaction. In addition, it is important to keep in mind that customers may be either internal or external. The external customer is the ultimate customer—that is, the project sponsor or client. However, internal customers are just as important and may be thought of as an individual or group who are the receivers of some project deliverable or an output of a process.

For example, project team members may be assigned the task of defining the detailed user requirements for an application system. These requirements may be handed off to one or several systems analysts who will develop the design models and then hand these models off to the programmers. The quality of the requirements specifications, in terms of accuracy, completeness, and understandability, for example, will have a direct bearing on the quality of the models developed by the systems analysts. In turn, the quality of the models will impact the quality of the programs developed. Therefore, we can view the series of project and software development processes as a customer chain made up of both internal and external customers,

As you might expect, a chain is only as strong as its weakest link, and any quality problems that occur can impact the quality of the project's product downstream. The primary focus of the project team should be to meet or exceed the expectations and needs of their customer because the customer is the ultimate judge of quality (Ginac 1998).

Prevention not Inspection One of Deming's most salient ideas is that quality cannot be inspected into a product. Quality is either built into the product or it is not. Therefore, the total cost of quality is equal to the sum of four components—prevention, inspection, internal failure, and external failure. The cost associated with prevention consists of all the actions a project team may take to prevent defects, mistakes, bugs, and so forth from occurring in the first place. The cost of inspection entails the costs associated with measuring, evaluating, and auditing the project processes and deliverables to ensure conformance to standards or requirement specifications. Costs of internal failure can be attributed to rework or fixing a defective product before it is delivered to the customer. These types of problems are, hopefully, found before the product is released. External failure costs entail the costs to fix problems or defects discovered after the product has been released. External failure costs can create the most damage for an organization because the customer's views and attitudes toward the organization may keep the customer from doing repeat business with the organization

in the future. Thus, prevention is the least expensive cost and can reduce the likelihood of a defect or bug reaching the customer undetected. In turn, this will reduce the cost of developing the system and improve the overall quality of the product (Lewis 2000).

Improve the Process to Improve the Product Processes are needed to create all of the project's deliverables and the final product—the information system. Subsequently, improving the process will improve the quality of the product. Project processes must be activities that add value to the overall customer chain. In addition, processes can be broken down into subprocesses and must be repeatable and measurable so that they can be controlled and improved. Improving any process, however, takes time because process improvement is often incremental.

Quality Is Everyone's Responsibility Quality improvement requires time and resources. As many of the quality gurus point out, quality has to be more than just a slogan. It requires a commitment from management and the people who will do the work. Management must not only provide resources, but also remove organizational barriers and provide leadership. On the other hand, those individuals who perform the work usually know their job better than their managers. These people are often the ones who have direct contact with the end customer. Therefore, they should be responsible and empowered for ensuring quality and encouraged to take pride in their work. Quality improvement may not be all that easy to achieve because it may require an organization to change its culture and focus on long-term gains at the expense and pressure to deliver short-term results.

Fact-Based Management It is also important that a quality program and project quality plan be based on hard evidence. As Kloopenborg and Petrick (2002) point out, managing by facts requires that the organization (1) capture data and analyze trends that determine what is actually true about its process performance, (2) structure itself in such a way that it is more responsive to all stakeholders, and (3) collect and analyze data and trends that will provide a key foundation for evaluating and improving processes.

Quality Standards and Metrics

Standards provide the foundation for any quality plan; however, standards must be meaningful and clearly defined in order to be relevant and useful. As illustrated in Figure 10.9, the project's goal, defined in terms of the measurable organizational value or MOV, provides the basis for defining the project's standards. The MOV defines the project's ultimate goal in terms of the explicit value the project will bring to the organization. In turn, the MOV provides a basis for defining and managing the project's scope, which defines the high-level deliverables of the project as well as the general features and functionality to be provided by the IT solution. However, the scope of the project, in terms of the features and functionality of the information system, are often defined in greater detail as part of the requirements definition.

As Figure 10.9 illustrates, the project's standards can be defined in terms of the project's deliverables and, most importantly, by the IT solution to be delivered. Once the features, functionality, or requirements are defined, the next step is to identify specific quality attributes or dimensions associated with each project deliverable. A customer-driven quality assurance plan first identifies each customer's requirements, represents them as quality attributes or dimensions, and then translates those dimensions into metrics (Ginac 1998). For example, Kan (1995) suggests several dimensions that can serve as quality standards for the software product. These include the application's features, reliability, usability, performance,

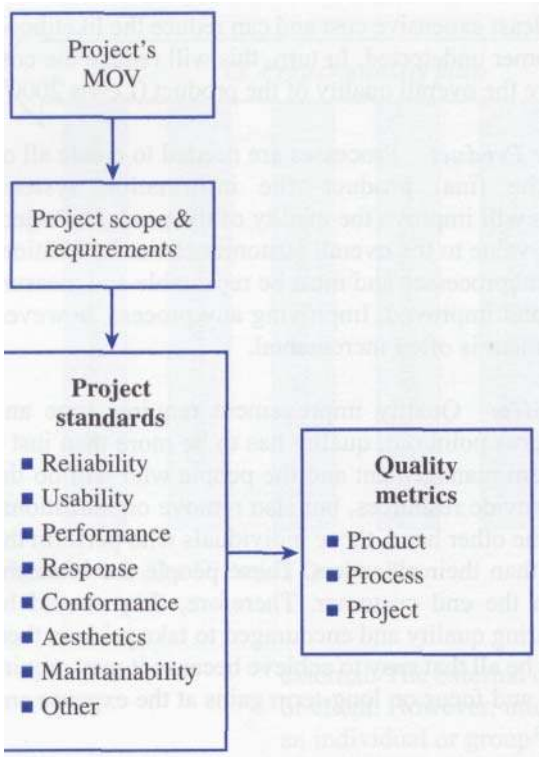


Figure 10.9 Developing Standards and Metrics

response, conformance, aesthetics, and maintainability. Although these dimensions focus on the application system, other dimensions can be identified for each of the project deliverables (e.g., business case, project charter and baseline project plan, project reporting, user documentation, etc.).

Metrics are vital for gauging quality by establishing tolerance limits and identifying defects. A **defect** is an undesirable behavior associated with the product or process (Ginac 1998). It is a failure to comply with a requirement (Lewis 2000). In software development, defects are often referred to as bugs.[1]

Once the quality dimensions are identified, the next step is to define a set of metrics that allow the project manager and team to monitor each of the project standards. There are two parts to a metric—the metric itself and an acceptable value or range of values for that metric (Ginac 1998). Metrics should focus on three categories (Kan 1995):

- *Process*—The control of defects introduced by the processes required to develop or create the project deliver ables. Process metrics can be used to improve software development or maintenance processes. Process metrics should focus on the effectiveness of identifying and removing defects or bugs.
- *Product*—The intrinsic quality of the deliverables and the satisfaction of the customer with these deliverables. These metrics should attempt to describe the characteristics of the project's deliverables and final product. Examples of product metrics may focus on customer sat isfaction, performance, reliability, and design features.
- *Project*—The control of the project management processes to ensure that the project meets its overall goal as well as its scope, schedule, and budget.

Metrics can be used to determine whether the software product and project deliverables meet requirements for "fitness for use" and "conformance to requirements" as defined by the internal or external customers. Many technical people, however, often feel that standards are restricting and only serve to stifle creativity. Although too many standards that are rigidly followed can lend support to that argument, well-defined standards and procedures are necessary for ensuring quality. A quality approach can also decrease development costs because the sooner a defect or bug is found and corrected, the less costly it will be down the road (Lewis 2000). Table 10.3 provides a summary of some common process, product, and project metrics.

[1] The term *bug* was introduced to the computer field by Dr. Grace Murray Hopper (1906—1992)—an extraordinary woman who retired as a Rear Admiral in the U. S. Navy. In 1946, while working on the Mark II and Mark III computers, she found that one of the computers crashed as a result of a moth that had became trapped in one of the computer's relays. The moth was carefully removed and taped to the logbook where an inscription was made that the computer was *debugged.* For some reason the term stuck, and errors, or *glitches,* in a program or computer system are called *bugs.*

Table 10.3 Examples of Process, Product, and Project Metrics

Type	*Metric*	*Description*
Process	Defect arrival rate	The number of defects found over a specific period of time
	Defects by phase	The number of defects found during each phase of the project
	Defect backlog	The number of defects waiting to be fixed The average time it
	Fix response time	takes to fix a defect The number of fixes that created new
	Defective fixes	defects
Product	Mean time to failure	Average or mean time elapsed until a product fails
	Defect density	The number of defects per lines of code (LOG) or function points
	Customer found defects	The number of defects found by the customer
	Customer satisfaction	An index to measure customer satisfaction—e.g., scale from 1 (very unsatisfied) to 5 (very satisfied)
Project	Scope change requests	The number of scope changes requested by the client or sponsor
	Scope change approvals	The number of scope changes that were approved
	Overdue tasks	The number of tasks that were started but not finished by the expected date or time
	Tasks that should have started	The number of tasks that should have started but have been delayed
	Over budgeted tasks	The number of tasks (and dollar amount) of tasks that have cost more to complete than expected
	Earned value	Budgeted Cost of Work Performed (BCWP)
	Over allocated resources	The number of resources assigned to more than one task
	Turnover	The number of project team members who quit or terminated
	Training hours	The number of training hours per project team member

Verification and Validation

Verification and validation (V&V) are becoming increasingly important concepts in software engineering (Jarvis and Crandall 1997). V&V activities continually prompt us to ask whether we will deliver an IT solution that meets or exceeds our project sponsor's expectations.

The concept of **verification** emerged about twenty years ago in the aerospace industry, where it is important that software perform all of its intended functions correctly and reliably because any error in a software program could result in an expensive or disastrous mission failure (Lewis 2000). Verification focuses on the process-related activities of the project to ensure that the product or deliverable meets its specified requirements before final testing of the system begins.

Verification requires that the standards and metrics be defined clearly. Moreover, verification activities focus on asking the question of whether we followed the right procedures and processes. In general, verification includes three types of reviews (Ginac 1998):

- *Technical Reviews*—A technical review ensures that the IT solution will conform to the specified requirements. This review may include conformance to graphical user interface (GUI) standards, programming and documentation standards, naming conventions, and so forth. Two common approaches to technical reviews include structured walkthroughs and inspections. A **walkthrough** is a review process in which the programmer or designer leads a group of programmers or designers through a program or technical design.

The participants may ask questions, make comments, or point out errors or violations of standards (Ginac 1998). Similarly, **inspections** are peer reviews in which the key feature is the use of a checklist to help identify errors. The checklists are updated after data is collected and may suggest that certain types of errors are occurring more or less frequently than in the past (Lewis 2000). Although walkthroughs and inspections have generally focused on the development of programs, they can be used as a verification of all project deliverables throughout the project life cycle.

Business Reviews—A business review is designed to ensure that the IT solution provides the required functionality specified in the project scope and requirements definition. However, business reviews can include all project deliverables to ensure that each deliverable (1) is complete, (2) provides the necessary information required for the next phase or process, (3) meets predefined standards, and (4) conforms to the project methodology.

Management Reviews—A management review basically compares the project's actual progress against the baseline project plan. In general, the project manager is responsible for presenting the project's progress to provide a clear idea of the project's current status. Issues may need to be resolved, resources adjusted, or decisions made to either stay or alter the project's course. In addition, management may review the project to determine if it meets the scope, schedule, budget, and quality objectives.

Validation, on the other hand, is a product-oriented activity that attempts to determine if the system or project deliverable meets the customer or client's expectations and ensures that the system performs as specified. Unlike verification, validation activities occur toward the end of the project or after the information system has been developed. Therefore, testing makes up the majority of validation activities. Table 10.4 provides a summary of the various types of tests that can be conducted for a software engineering project. Volumes and courses can be devoted to software testing, so just an overview (or refresher) can be provided in this text. However, understanding what needs to be tested and how is an important consideration for developing a quality strategy and plan for the IT project.

Table 10.4 Testing Approaches

Test	*Description*
Unit testing	Unit testing is done at the module, program, or object level and focuses on whether specific functions work properly. Unit testing can be accomplished via: • *Black box testing*—Tests the program code against specified requirements (i.e., functionality) • *White box testing*—Examines paths of logic inside the program (i.e., structure) • *Gray box testing*—Study the requirements and communicate with the developer to understand internal structure of the program (i.e., functionality and structure)
Integration testing	Tests whether a set of logically related units (e.g., functions, modules, programs, objects, etc.) work together properly after unit testing is complete
Systems testing	The system is tested as a whole in an operating environment to verify functionality and fitness for use. May include tests to verify usability, performance, stress, compatibility, and documentation
Acceptance testing	To certify that the system satisfies the end customer's scope and detailed requirement specifications after systems testing is complete. The end user or client is responsible for assuring that all specified functionality is included and will provide value to the organization as defined by the project's goal or MOV.

Testing provides a basis for ensuring that the system functions as intended and has all the capabilities and features that were defined in project's scope and requirements. In addition, testing provides a formal, structured, and traceable process that gives management and the project

sponsor confidence in the quality of the system (Lewis 2000). In addition, Lewis (2000) provides several suggestions for making software testing more effective:

- Testing should be conducted by someone who does not have a personal stake in the project. In other words, programmers should not test their own pro grams because it is difficult for people to be objective about their own work.
- Testing should be continuous and conducted throughout all the develop ment phases.
- In order to determine whether the test met its objectives correctly, a test plan should outline what is to be tested, how it will be tested, when it will be tested, who will do the testing, and the expected results.
- A test plan should act as a service level agreement among the various proj ect stakeholders and should encourage "quality before design and coding."
- A key to testing is having the right attitude. Testers should not be out to "break the code" or embarrass a project team member. A tester should eval uate a software product with the intent of helping the developers meet the customer's requirements and make the product even better.

Change Control and Configuration Management

Suppose you were developing a database application system for a client. After several weeks, you would undoubtedly make a number changes to the tables, attributes, user interface, and reports as part of a natural evolution of the project. This evolution is both normal and expected as you learn more about the technology and the requirements. In addition, the user/client may suggest changes or enhancements if the organizational environment changes.

If you are working alone, you may store all the products of the software development (i.e., reports, plans, design models, program and database files) on your computer. Change control may be nothing more than just keeping your documents and files organized. If, however, you need to share these files and documents with even one other person, controlling these changes becomes more problematic. You could all keep the files and documents being worked on at everyone's stand-alone workstation. Unfortunately, if you need to share or work on the same documents or files, this sharing can lead to several different versions of the same document or file distributed among several different computers. On the other hand, you may store all the work in a shared directory on a server. This solution would certainly allow everyone to share and use the same documents or files, but problems could occur if two or more people work on the same document or file at the same time. The changes one makes would be lost if someone else were to save a file after the first person saved it, thus replacing new file with a different new file. There could be a great deal of confusion and wasted time.

Change is inevitable throughout the life of the project. On any given project, each deliverable will progress through a series of stages from an initial conception through a final release. As the deliverable develops, changes will be made informally until it gets to a state of completeness, whereupon revision control is needed. At some point informal changes should be no longer permitted. After final acceptance, the deliverable should be frozen until it is released. An informal change control allows changes that can be traced and captured sequentially to be made to an evolving project deliverable. It provides for rapid development while allowing for backup and some measures of control. On the other hand, formal change control is a procedure in which changes to an accepted work are formally proposed and assessed and decisions to accept or reject proposed changes are documented to provide an element of stability beyond the informal change controls.

Configuration management is an important aspect of PQM that helps control and manage document and software product change (Jarvis and Crandall 1997). It provides the project team with an environment for efficiently accessing different versions of past documents or files. Its basic purpose is to establish and maintain the integrity of the various project work products and deliverables throughout the project life cycle. In short, configuration management attempts to answer the following basic questions (Ginac 1998):

- What changes were made?
- Who made the changes?
- When were the changes made?
- Why were the changes made?

Configuration management tools allow different project team members to work on a specific section of a document or file. The document or file can be checked out and checked back into a repository or library in order to maintain control. Software and the supporting project deliverables often go through an evolution of successive temporary states called versions (Lewis 2000). Configuration management, therefore, includes a set of processes and tools that allows the project team to manage its various documents and files as various configurations of IT solutions and project deliverables are derived. It may include specifying and enforcing various policies that restrict access to specific individuals or preventing two people from changing the same document or file at the same time (Ginac 1998).

According to Lewis (2000), software configuration management includes four elements—component identification, version control, configuration building, and change control.

Component Identification This first element focuses on the processes or activities for defining or describing the various software configuration items or work products that make up a specific project deliverable. Guidelines are established and followed for identifying and naming the various baselines, software components, and configurations. As these elements go through changes, a numbering and/or naming scheme is used to uniquely identify each of the various versions or revisions as they evolve and change over time. The various components are often stored in a library or repository, where a list of all the components can be cataloged.

Version Control As the project deliverables and work products evolve and change over time, many different versions are created. Errors may be corrected and enhancements are made until the work product becomes stable. Each evolutionary change results in a new version. It is essential that these components be organized so that different versions can be distinguished from one another. With the exception of the first version, each subsequent version will have a predecessor and the ability to trace each version becomes the component's history. Allowing the project team to go back to any single version provides an important backup and allows for specific ideas to be saved and made available for reuse later on.

Configuration Building Configuration building entails identifying the correct component versions and then being able to execute the build procedures. A **build** includes all the software components, such as data files, programs, and so forth that are needed to implement one or more software functions (Pressman 2001). A software product must be built in order for it to run. For example, if you have a single program,

building the application may require compiling and linking the program file in order to create an executable program. However, a larger application system may require hundreds or even thousands of files to be compiled, linked, and combined to create an executable system. This process can become time-consuming and complicated (McConnell 1996). Therefore, configuration building ensures that the derived software components are correctly associated and put together with each other in order to create an accurate build.

Change Control Once a software component becomes stable and accepted, a decision process must be in place to control any proposed changes. Moreover, a simple change will often involve several other components, so it is important that the impact of any change requests be assessed. The change control activities ensure that any modification to a software component is proposed, evaluated, approved or rejected, scheduled, and tracked. It provides the basis for reporting and auditing processes. If a change is made, the component should be checked back into the library or repository where it becomes a new component version and the previous version is retained.

Monitor and Control

Quality control focuses on monitoring the activities and results of the project to ensure that the project complies with the quality standards. Once the project's standards are in place, it is important to monitor them to ensure that the project quality objective is achieved. Moreover, control is essential for identifying problems in order to take corrective action and also to make improvements once a process is under control.

Similar to the quality assurance activities, quality control should be ongoing throughout the life cycle of the project and only end when the customer or project sponsor accepts the final IT solution (Kloppenborg and Petrick 2002). Moreover, quality control includes monitoring and controlling activities concerning the product, processes, and project. Using the system concept as illustrated in Figure 10.10, quality control activities must focus on the inputs and outputs of each process. If inputs to a process are of poor quality, then the output of a particular process will be of poor quality as well because, in general, the process may not be capable of changing the inherent quality of the input. Moreover, even if the input to a process is of high quality, the process itself may create an output of lower quality. Finally, the input and process may not produce a quality output or product if the requirements are not properly defined.

To support the quality control activities, several tools and techniques were introduced in this chapter. Figure 10.11 provides a summary of those tools. As Besterfield, et al (1999) point out, these tools can be used to monitor the process, product, and product metrics in order to:

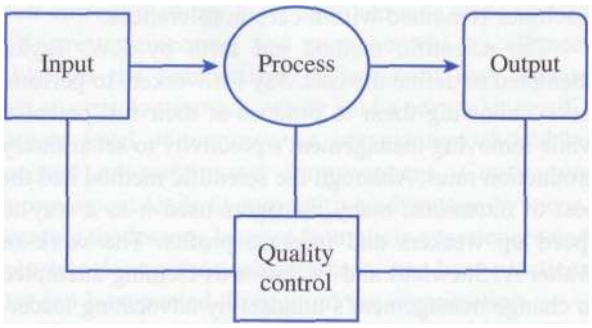
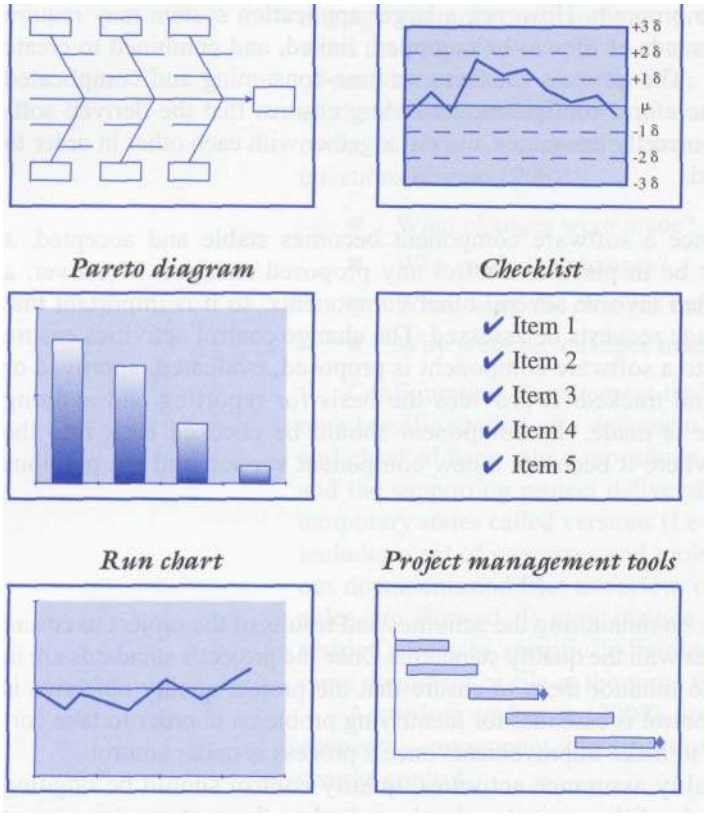


Figure 10.10 Quality Control Activities

Learn, Mature, and Improve

A central theme of this text has been the application of knowledge management as a tool for team learning and identifying best practices. Monitoring and controlling activities and tools can help point out problem areas, but the project team must solve these problems. Therefore, it is important that the lessons



Cause and effect diagram *Control chart*

Figure 10.11 Quality Control Tools

learned from a project team's experiences be documented so that best practices be identified and disseminated to other project teams. Continual, incremental improvements can make a process more efficient, effective, stable, mature, and adaptable (Besterfield, Besterfield-Michna et al. 1999). A project quality plan should be more than an attempt to build a better IT solution, it should also support the organization in searching for ways to build a better product (Woodall, Rebuck et al. 1997).

CHAPTER SUMMARY

Project quality management (PQM) is a knowledge area defined by the Project Management Body of Knowledge. It is defined as:

the processes to ensure that the project will satisfy the needs for which it was undertaken. It includes all activities of the overall management function that determine the quality policy, objectives, and responsibility and implements them by means of quality planning, quality assurance, quality control, and quality improvement, within the quality system.

In this text, PQM has been expanded to include not only the quality management concepts, but also verification and validation activities and change control to manage the various configurations of the project products throughout the project life cycle.

Although quality can mean different things to different people, quality in organizational settings has been traditionally defined as "fitness for use" and "conformance to requirements." Before the focus on interchangeable parts, quality was controlled by guilds that regulated membership, pricing, and trade in a particular town. With Eli Whitney's concept of mass producing interchangeable parts as part of a *manufactory*, the seed for the modern assembly line was born, Instead of training people to perform skilled work, they could instead be trained to operate machines to do the work, as long as the parts produced by the machines remained within certain tolerances,

The scientific method put forth by F.W Taylor attempted to define the best way for workers to perform tasks—allowing them to produce at their full potential whlle **removing** management's proclivity to set arbitrary production rates. Although the scientific method had the best of intentions, many managers used it as a way to speed up workers and increase profits. The work of Walter A. Shewhart and W. Edwards Deming attempted to change management's mindset by advocating leadership, prevention over inspection, and statistical control to improve productivity and quality. Because Japan faced

the daunting task of rebuilding its economy after World War II with few natural resources and a reputation for inferior goods, a group called the Union of Japanese Scientists and Engineers (JUSE) was formed with the help of Japan's allies to help transform the nation. As part of this effort, Deming and Joseph Juran were invited to give lectures on statistical quality control. Japanese managers embraced these principles and ideas, and the quality movement was officially bora. Many others, such as Kaoru Ishikawa and Philip Crosby, contributed to this worldwide movement, and proprietary and nonpropri-etary quality management systems have gained increasing popularity in many organizations.

As part of the quality movement, standards in the form of documented agreements, protocols, or rules that outline specific criteria for quality became the backbone for ensuring quality. Several organizations and quality initiatives have gained fame over the years. ISO, probably the most widely known standards organization, was formed in 1947 with the intention of creating and coordinating a set of international standards. While the ISO 14000 focus on environmental management, the ISO 9000 focus on eight quality management principles that provide a framework for different organizations. A third party, called a registrar, can audit an organization and issue a certification that the organization's processes conform to the ISO standards.

Other quality initiatives, such as Six Sigma, focus on variations in processes that may translate into products or services that do not meet customer needs and expectations. By improving the quality of its processes, an organization can achieve its Six Sigma goal of only producing 3.4 defects per million. More recently, the Software Engineering Institute at Carnegie Mellon University introduced the Capability Maturity Model (CMM) that provides a set of recommended practices for a set of key process areas specific to software development. The CMM also provides a path of five levels to help organizations determine their current maturity level and then take steps toward software engineering and management excellence. Although the competitive environment may dictate that an organization achieve or hold a particular certificate or level of maturity, an organization should be focused on continuous improvement. Continuous improvement leads to competitive advantage by incorporating the lessons learned from their experiences and then translating those experiences into best practices that can be repeated throughout the organization.

The concepts, tools, methods, and philosophies of the quality movement provide a foundation for developing the IT project quality plan. The plan should be based on the following:

- *Quality Philosophies and Principles*—To guide the plan's objective and mission.
- *Quality Standards and Metrics*—To define the quality objectives and expectations and to provide a baseline for benchmarking improvements.
- *Validation and Verification Activities*—To ensure a quality approach throughout the project. Verification activities, such as technical, business, and management reviews, determine whether the project team is building the system or producing project deliverables according to specified stan dards or requirements; validation activities, such as software testing, tend to focus on whether the pro ject's products will meet customer expectations.
- *Change Control and Configuration Management*—To support the natural evolution of the project's products. As these products evolve, change is inevitable. It is important that this change is man aged effectively in order to reduce confusion and wasted effort. It includes a document repository library where files or documents can be checked out and checked in as needed. This process allows for versioning, backup, and safeguarding so that docu ments or files are not accidentally replaced by other project team members. Configuration building also allows for identifying the correct component ver sions needed to execute build procedures. Configuration management also provides formal change control to ensure that changes to accepted work are formally proposed and assessed and any decisions to make the changes are documented.
- *Monitor and Control*—To focus on monitoring the project activities to ensure that the project meets its quality standards. Once the project work begins, it is important that these activities be monitored and assessed so that appropriate corrective action can be taken when necessary. Quality control tools and techniques can be used to monitor each project or software development process and the inputs and outputs of the process, as well.
- *Learn, Mature, and Improve*—To focus on continu ous quality improvement. As a project progresses, lessons learned can be documented from the project team's experiences. Recommendations, issues, chal lenges, and opportunities can be identified and shared with other project teams; and many of these experi ences can provide the basis for best practices that can be implemented throughout the organization.

WEB SITES TO VISIT

Quality Gurus
http: //w w w.j uran.com/
http://www.deming.org/
http://www.philipcrosby.com/

UCITA
http://www.infoworld.com/ucita/

ISO
http://www.iso.ch/iso/en/ISOOnline.frontpage
Software Engineering Institute/ CMM
http://www.sei.cmu.edu/ *Configuration Management* http://www.cmtoday.com/

REVIEW QUESTIONS

1. Define quality in your own words. How would you define quality in a word processing, spreadsheet, or presentation software package?
2. Why is the number of features of a software system not necessarily the best measure of that system's quality?
3. How does "conformance to requirements" or "fit ness for use" provide a definition of quality for an information system or software product?
4. What is PQM?
5. Define the following: (a) Quality Planning; (b) Quality Assurance; (c) Quality Control
6. Why should quality management include both the products and processes of a project?
7. What is scientific management? Why was it so pop ular? Why was it so controversial?
8. What is a control chart? When is a process said to be in statistical control? How would you know if it was not?
9. Why did the teachings of Deming and Juran have such an important impact on Japan just after World War II?
10. What is an Ishikawa diagram? How can it be used as a quality control tool for an IT project?
11. What is a Pareto diagram? How can it be used as a quality control tool for an IT project?
12. What is a flow chart? How can it be used as a qual ity control tool for an IT project?
13. What is a standard? What role do standards play in developing an information system?
14. What is ISO? Why would an organization wish to be ISO certified?
15. What is the difference between ISO 9000 and ISO 14000?
16. Can an organization be ISO compliant but not cer tified?
17. What is TickIT?
18. Briefly describe Six Sigma and its objectives.
19. How does achieving a Six Sigma objective improve quality?
20. What is process capability?
21. What is process maturity?
22. Describe an immature software organization.
23. Describe a mature software organization.
24. What is the relationship between standards and metrics?
25. What is a process metric? Give an example.
26. What is a product metric? Give an example.
27. What is a project metric? Give an example.
28. What is a defect? Give an example of a software defect.
29. Describe verification. What activities support veri fication?
30. Describe validation. What activities support vali dation?
31. Describe how technical, management, and business reviews are different.
32. What is the purpose of change control?
33. Why should some changes be allowed to be made informally, while other changes should be made formally?
34. What is configuration management? How does it support change control?
35. What role does knowledge management play in continuous quality improvement?

EXTEND YOUR KNOWLEDGE

1. Interview two or three people who regularly use an application software package. Examples of an appli cation software package include an Internet browser, electronic spreadsheet package, or a word processing package. Summarize each interview in one or two pages based upon the following questions:
   a. What application software package do you use the most?
   b. How often do you use this particular software package?
   c. Which features or functions do use the most? The least?
   d. How would you rate the overall quality of the software package on a scale from one to ten, where one indicates very low quality and ten indicates very high quality?
   e. Why did you give the software package this score?
   f. In your opinion, what are the three most important attributes of a high quality software package?
2. Contact someone in an organization who is willing to talk to you about her experiences implementing a quality program such as Six Sigma, ISO, Ticklt, or the CMM. If this is not feasible, use the Internet or library to find an article. Prepare a short report that answers the following:
   a. What were the compelling reasons for initiating a quality program?
   b. What was the biggest challenge that the organi zation faced when trying to implement the qual ity program?
   c. How long did it take to implement the program? Or how far along are they?
   d. What lessons did the organization learn from its experience?
3. You and two other students have been hired by a local swim team to develop a Web site that will provide information about the team. The information on the Web site will be used to recruit new swimmers and will provide information to current members about upcoming meets and practices. In addition, team and individual statistics will be posted after each swim meet. Before you begin, you need to develop a quality plan. The plan should include:
   a. Your own quality philosophy.
   b. Two metrics for ensuring that reliability stan dards are met.
   c. Two metrics for ensuring that performance stan dards are met.
   d. A means for validating and verifying that your client's needs and expectations will be met.

BIBLIOGRAPHY

Besterfield, D. H., C. Besterfield-Michna, et al. 1999. *Total Quality Management.* Upper Saddle River, N.J.: Prentice Hall. Boehm, B. W. 1981. *Software Engineering Economics.* Englewood Cliffs, N.J.: Prentice Hall. Caputo, K. 1998. *CMM Implementation Guide: Choreographing Software Process Development.* Reading, Mass.: Addison-Wesley. Deming, W. E. 1982. *Out of the Crisis.* Cambridge, Mass.: The MIT Press. Florae, W. A., R. E. T. Park, et al. 1997. *Practical Software Measurement: Measuring for Process Management and Improvement.* Pittsburgh, Pa.: Software Engineering Institute. Ginac, F. P. 1998. *Customer Oriented Software Quality Assurance.* Upper Saddle River, N.J.: Prentice Hall. Humphrey, W. 1988. Characterizing the Software Process: A Maturity Framework. *IEEE Software* 5(3): 73-79. Jarvis, A. and V. Crandall. 1997. *Inroads to Software Quality: How to Guide and Toolkit.* Upper Saddle River, N.J.: Prentice Hall PTR. Kan, S. H. 1995. *Metrics and Models in Software Quality Engineering.* Boston, MA: Addison-Wesley.

Kloppenborg, T. J. and J. A. Petrick. 2002. *Managing Project Quality.* Vienna, VA: Management Concepts. Lewis, W. E. 2000. *Software Testing and Continuous Quality Improvement.* Boca Raton, FL: Auerbach. McConnell, S. 1996. *Rapid Development: Taming Wild Software Schedules.* Redmond, WA: Microsoft Press. Paulk, M. C. 1994. A Comparison of ISO 9001 and the Capability Maturity Model for Software. *Software Engineering Institute* CMU/SEI-94-TR-12. Paulk, M. C., B. Curtis, et al. 1993. The Capability Maturity Model for Software. *IEEE Software* 10(4): 18-27. Pressman, R. S. 2001. *Software Engineering: A Practitioner's Approach.* Boston, MA, McGraw-Hill. Pyzdek, T. 1999. *The Complete Guide to Six Sigma.* Quality Publishing.

Siviy, J. 2001. *Six Sigma.* The Software Engineering Institute (SEI).

Williamson, M. 1997. Quality Pays. *Computerworld* (August 18).

Woodall, J., D. K. Rebuck, et al. 1997. *Total Quality in Information Systems and Technology.* Delray Beach, FL: St. Lucie Press.