Title: Software Engineering and TQM.

Course:
Year: 1994
Author(s): F. Rivera

Report No: P94064

Abstract: Quality cost, quality management, quality assurance, quality control, quality improvement, personnel for quality, employee involvement, and statistical process control are all issues in TQM that can be implemented in Software Engineering Organizations. In this research, the author tries to show different techniques, methodologies and tools that today are in use by many organizations to address the necessity of excellence in software development. In order to organize the work, the author has divided it in three sections. The section Team Building presents a framework to help Information System Managers ISM) to build a software team. The second is a series of techniques and methodologies that can be adopted to Fix the Process through standardization, continuous improvement, bench-marking, statistics process control, and quality assurance. The third, the Voice of the Customer, is an integration of software development life cycle and customer satisfaction.

Software Engineering and TQM

Fernando E. Rivera

EMP-P9464

RESEARCH PAPER

EMGT 510 TQM - I

SOFTWARE ENGINEERING AND TQM

PORTLAND STATE UNIVERSITY

MARCH 8, 1994

FERNANDO E. RIVERA

Software Engineering and TQM

Introduction

Quality cost, quality management, quality assurance, quality control, quality improvement, personnel for quality, employee involvement, and statistical process control are all issues in TQM that can be implemented in Software Engineering Organizations.

In this research I will try to show different techniques, methodologies and tools that today are in use by many organizations to address the necessity of excellence in software development.

In order to organize the work I had divide it in three sections. The first section Team Building presents a framework to help Information System Managers (ISM) to build a software team. The second a series of techniques and methodologies that can be adopted to Fix the Process through standardization, continuous improvement, benchmarking, statistics process control, and quality assurance. The third The Voice of The Customer is an integration of software development's life cycle and customer satisfaction.

Team Building

Every project should start with an informal but detailed statement of known requirement. Rettig and Simonds[1] suggest to divide this statement in four parts: 1) Project title, the name the entire organization will use to describe the project; 2) Purpose, a concise statement of the specific goals of the project; 3) Stages, a numbered list of the development stages the project will go through on its way to completion (schedule) and 4) The team, people assigned to the project team, with role assignment.

The first two statements have only the purpose of distinguish the project from others within the organization and generally do not represent a major activity. The third refers to the common belief that small projects are easier to manage than big projects and is a general practice in the software industry (top-down decomposition). However, the last one represent one of the major problems to managers: How to build the team?

Constantine[2] introduces a theorical framework that defines four approaches to build a technological team: Closed, Random, Open, and Synchronous. They are based, respectively, on traditional hierarchy of authority, on creative independent initiative, on adaptive collaboration and communication, and on alignment with a shared vision. This framework can provide a tool for building teams in software organizations.

---

[1] Rettig, M. and Simonds, G., "A Project Planning and Development Process for Small Teams", *Communications of ACM*, October 1993, v36, n10, p45.
[2] Constantine, L., "Building structured open teams to work", *Software Development 1991 Proceedings*. Miller Freeman, San Francisco, California, 1991.

Closed

Such organization is structured as pyramid or hierarchy with distinct and well-defined roles specified for each position in the hierarchy. People assume that someone has to be in charge of others and that decision must be made by whoever is in charge. Information is controlled and channeled along lines of authority and decision made by managers. Corporate or collective interests come first. Individuals are expected to demonstrate loyalty and deferred to the group. Insubordination is not tolerated, and opposition or criticism may be seen as disloyal. Within this scheme, standards and rules of operation promote continuity instead of innovation. The advantages are in stability and predictable performance.

Rettig and Simonds definition for a hierarchical project team consist at minimum of three roles: 1) Implementer, a programmer who writes the program code; 2) Domain specialist, a specialist in the application domain and 3) Technical reviewer, a programmer who serves to review technical aspects of the implementation. In large project, two or more members are designated for each role. Besides the above role, one member might play the role of team leader or someone can be nominated to play solely this position. The team leader has the following responsibilities:

Facilitator: Schedule and lead team meetings.

Archivist: Take minutes of team's meetings and maintain project's documentation.

Manager: Maintain a char of progress.

Contact: Serve as contact point between the project team and the rest of the organization.

Random

The random approach is the antithesis of the closed. In random the team might rely on individual initiative as the basis of decisions. The freedom of the individual to create and act independently is considered more important than group interests. The random organization excel at creative invention but are not highly stable or efficient.

Open

The open approach has characteristics from the closed and random. It is based on adaptive collaboration, integrating innovation with stability and individual with collective interests through negotiation and discussion. This is a model which roles and responsibilities are flexibly shared. Because groups based on this approach share information so freely and combine diverse approaches, they excel at solving complex problems. However, they can waste too much time in fruitless debate.

Synchronous

The synchronous approach is the opposite of the open. It is based on harmonious and effortless coordination through the alignment of members with a common vision that reflect the collective goals and methods of reaching those goals. In such groups, smooth, efficient operation with quiet unanimity is all important. Synchronous groups can be efficient in smoothly performing established procedure, yet they tend not to be highly responsive or adaptive to changing requirements.

Project teams can be based on any of the above approach, but which is the ideal for software development project?

Software projects typically involve a combination of complex problem solving with the need for a certain amount of innovation that traditionally has been addressed by individual programmers or hierarchical teams. Zahniser[3] think that today it is necessary to make a paradigm shift from expert knowledge worker performing fairly independently to produce modular system component to cross-functional teams of various system experts working together to produce a consensus system solution to a jointly defined user-problem set. Collaborative teamwork and consensus engineering have many characteristics that make them appealing as the basis for the software development. The Open approach team seems to be the in advantage to share and utilize complex information and integrating the contributions of all team's members into a single, practical high-quality solution. Constantine proposes a hybrid named 'structured open' that combine *closed* (hierarchical) elements and *open* ones with some of *random* teamwork. It uses hierarchical structures to promote flexibility along with more efficient group problem solving. For high performance certain key group roles are identified, but instead of to be assigned permanently to a specific member, these are collectively shared by the entire team and rotated among the members. In this way the model promotes consensus building with high level of participation and the diffusion of skill among the members.

Choosing the team members

In an organization pursuing excellence the members need to be committed, focuses on the customer able to communicate and dedicated to continuous improvement. Programmers

[3] Zahniser, Richard A., "TQM for Technical Teams", *Communications of ACM*, October 1993, v36, n10, p. 115.

rarely make the best documents or trainers. Trainers usually do not like to program. Programmers rarely make good managers, and good managers are often pretty terrible programmers.

Depending of the needs and objectives of the organization different project teams can be build. Managers have to identify potential strength and weakness of the members and combine with the team model to achieve the highest performance.

People who do best in tactical teams (closed) have been described as loyal, committed and action oriented. They have a strong sense of urgency and respond well to leader ship.
People who work best within the creative environment of a breakthrough (random) team are independent thinker, often artistic or intellectual. They are persevering self-starters who do not need orders to get going or close supervision to keep going.
People who thrive in the collaborative consensus-building of problem-solving teams (open) are practical-minded but sensitive to people-issues. They have integrity and are seen as trustworthy by peers, exhibiting intelligence coupled with good interpersonal skill.
People that fit in synchronous teams are intuitive, somewhat introverted, yet people-sensitive. They are good at linking the large picture to specific action and work with quiet efficiency.

Fix the Process

Pitman estimate that poor quality cost 20%-50%[4] of operating expenses. This includes the costs of rework, paperwork, handling complains, failure-related costs, and lost future business. He says too that fixes that are necessary to make to correct requirements' definition errors frequently cost 100 times more to correct after software is installed that they would have if the requirements' definition had been defect-free. Costs of delivering quality include prevention costs, detection costs, and correction-before-installation costs.

Eight different techniques and methodologies are presented to address problems of quality cost, standardization, continuous improvement, productivity, and cultural change. Each one has its one's strength and weakness and can be applied to address more than one problem or in combination. The implementation of them generally will be unique to each organization.

Support Teams (ST)

ST can make software development more efficient by standardizing design, producing effective methodologies, determining user's needs and providing information[5]. ST can produce and maintain design standards, naming conventions and coding conventions. The teams can then ensure that everyone in the design process knows the standards. This approach should make standards easier to follow.

---

[4] Pitman, B., "Total Quality Management for Information Services", *Journal of Systems Management*, v43, i7, July 1992, p. 18.

[5] Comaford, C., "Expediting development with Support Teams", *PC Week*, March 15 1993, v10, p. 68.

Application Development Centers (ADC)

ADC consist in concentrate computer programming expertise and equipment at a single location in order to provide an organization with evaluation, guidance and implementation of new development tools and methodology[6]. The key idea is to promote changes in the organization with the guidance of a leader that must have a definite vision of how to proceed. The leader has to work improving morale and motivating staff while making sure their teams are recognized internally and trained in new technologies.

Reengineering

Reengineering is increasingly attracting the attention of ISM who are under pressure to reduce or control operating cost. Part of the attraction of Reengineering is that it can be performed by large corporations and small companies alike. However, Reengineering, requires long-term implementation in order to yield savings. Some companies are using the Software Process Program (SPP) at the Software Engineering Institute at Carnegie Mellon University[7] to help them bring their software Reengineering, projects on time and under budget. The program's review process helps evaluate and implement an organization's project planning, testing standards implementation an other process.

---

[6] Ray, G., "Development centers battle to get it done", *Computerworld*, August 24 1992, p. 77.
[7] Wexler, J., "Re-engineering report card", *Computerworld*, July 6 1992 v26, p. 51.

Software Quality Assurance (SQA)

One of the main aspects of TQM is 'defect free products'. SQA is the function in software organizations responsible for auditing the quality of the "line production" (programmers) and they have to alert managers of any deviations. But, SQA is still an evolving discipline. Professionals are still defining what SQA is, how it is performed, when it is needed on a project, and who performs it. Humpehreys[8] recommend to ISM to keep SQA out of the development process due to SQA "is concerned with auditing the process to ensure proper implementation" rather than with the process its self. By the other hand, Listons[9] recommend maintain the SQA people and the programmers together in one team.
By properly implementing an SQA program, software can be a defect free product[10].

Benchmarking

Benchmarking can be focused on two areas, internal and external benchmarks. Depending on the focus, different objectives are achieved. Internal benchmarks are used to drive continuous improvements for quality and productivity as well as predictions of outcomes. External benchmarks identify competitive standing and improvement's opportunities. As complementary activities, the external benchmark is used to identify new process requirements and the internal benchmark is used to focus the improvement efforts. Using

---

[8] Humphrey, W., *Managing the Software Process*, Addison-Wesley 1989, p. 432.
[9] Liston, B.,"TQM and Software Engineering: A personal Perspective", *The Fourth Symposium on Quality Function Deployment*, 1992, p. 10.
[10] Lowe, J. and Jensen, B., "Customer service approach to software quality", *Annual Quality Congress Transactions*, v46 1992, ASQC Milwaukee, WI USA. pp. 1077-1083.

benchmarks in the proper context allows an enterprise to make effective decision on their software development and delivery process[11].

Capability Maturity Model (CMM)

In developing applications, the CMM developed by the Software Engineering Institute, attempts to define a software developer process maturity. Using the idea of TQM, CMM helps developers in creating systems of control. At key points in the process, characteristics of quality are measured under a Statistic Process Control (SPC). CMM has a five-level rating system that emphasizes continuous improvement. The five levels of maturity that an organization can achieve are initial, repeatable, defined, managed, and optimizing[12]. In the first level, developers work in their own fashion without following any of the corporate knowledge or guidance. The second level features stable planning and product baseline. By level 3, defined, the team is using defined and institutionalized process to provide quality control. At level 4, the team is able to measure the software process through planning and tracking. In level 5, optimizing, the mature team continues to work on its process improvement. An interesting feature of the CMM is that can be used in a self-assessment, that mean no auditors are requires to certify quality, for mature organizations. CMM was designed to help save money and time, and to do a better job the first time.

---

[11] Krick, K., "Balancing the benchmark opportunities", *IEEE Global Telecommunications Conference and Exhibitions*, 1991, v1, pp. 167-169.

[12] Polinsky, S., "CMM can help you manage the process of application development", *InfoWorld*, Nov. 2 1992, p. 47.

ISO 9000-3

The ISO 9000-3 standard provides software developers with special interpretation if the International Organization for Standardization's IS 9000 standard for quality and consistency in procedures. Unlike many manufacturing production procedures, software processes are inexact and rarely fall into predictable procedures. However today some software firms have reached ISO 9001 certification under the ISO 9000-3 guidelines[13]. ISO 9000's extensive document control requirements are especially challenging to software companies because they often perform hundreds of revisions to programs or individual modules. Software test procedures cover a broad, undefined range, including test for functionality, file-handling, platform portability, and other features. However, in ISO 9001, support for continuous improvement is almost absent.[14]. An ISO 9001 certification actually tells little about its software-development capability. Certification means only that some basic practices are in place; in CMM terms, this translates to many *repeatable* levels, and some at *defined* level. Another weakness of ISO 9001 is that require external auditors.

Computer Aided Software Engineering (CASE)

Some companies are using CASE tools to simplify the process of software production and increase programmers' productivity. CASE offers several advantages including improvements in software quality and design, and in development productivity. However, its implementation is complex due to the necessary involvement non technical players like auditors that play a significant role in the review of systems. Another weakness if the lack

[13] According to *The Economist*, Jan. 23 1993 p. 79, almost 200 software firm have reached ISO 9001.
[14] Coallier, F., "How ISO 90001 fits into The Software World", *IEEE Software*, January 1994, p. 98.

of capacity to identify and manage costs[15], and the lack of industry standards to provide integration mechanisms and facilities to communicate and share data with the rest of the quality environment[16]

[15] DeBrandt, J., "Estimating maintenance costs", *Computerworld*, March 25 1991 v25, p. 69.

[16] Boudjlida, N. and Bassons, H., "Integration mechanisms in ALF", *Proceedings of the Second International Conference on Systems Integration - ICSI '92*, pp. 315-324.

The Voice of the Customer

Customer satisfaction

Traditional software development methodologies are focus on detecting errors by appraisal (reviews, inspections and testing). With these methodologies, the best that we can get is zero defects. Modern methodologies, like quality function deployment[17,18,19] (QFD), have been developed to add value to the design process.

With QFD we can concentrate on maximizing customer satisfaction from the software engineering process. In addition to SPC, which helps to avoid customer dissatisfaction, we can use QFD first to design, then to develop or code and finally, market software that customers want to buy. In this process we have to understand what is valuable to the customer, and deploy that understanding of value through the software life cycle.

Customers accept or buy software for the following reasons. It helps them to: 1) solve problems, 2) evaluate opportunities, 3) look good, or 4) image[20]. Any software that does not satisfied a customer in at least one of these four ways is valueless.

To satisfy customers, teams must focus on the software elements from the customer's point of view. This requires, first, understanding the customer's problems and opportunities developing technical requirements. There are three types of requirements: normal

---

[17] Hauser, John R. and Clausing, Don, "The House of Quality", *Harvard Business Review*, May-June 1988, p. 63.
[18] Evans, James R. and Lindsay, Williams M., *The Management and Control of Quality* (West Publishing Company, 1993) pp. 150-163.
[19] Hutchinson, C. and Kihara, T., "QFD as a Software Design Tool for Software Development", *The Fourth Symposium On Quality Function Deployment*, 1992, pp. 1-13.
[20] Zultern, Richard E., "TQM for Technical Teams", *Communications of ACM*, October 1993, v36, n10, pp. 79-91.

requirements (or needs), expected requirements, and exciting requirements[21]. Normal requirements are these typically collected by surveying or interviewing customers and expressed as "wants". The customer satisfaction is directly proportional to their presence. Expected requirements are these so obvious that nobody made mention. They do not satisfy but, meets expectations. Finally, exciting requirements are these unexpected. If there are not present, they do not dissatisfy, but if they are, they highly satisfy because they are beyond customer's expectation.

As customers generally do not provide expected and exciting requirements, the team have the responsibility of understand what are the problem and opportunities of the customers. Getting close to customers, using brainstorming and successively asking Why?[22], a complete set of requirements can be prepared including quality requirements, functional requirement, data requirement, and performance requirement. For example, "I want to have automatic files' back-up" is a quality requirement that includes data requirement, "files". "Short cuts must be provided to options in the first level" is a functional requirement and "The update must be on-line" is a performance requirement.

Then the software can be specified in term these requirements, developing successively hierarchical decomposition and deploying the requirements through the rest of the process. Once this task is done, the design stage must be achieve. Then, the software must be coded and implemented.

---

[21] Evans, James R. and Lindsay, Williams M., *The Management and Control of Quality* (West-Publishing Company, 1993) pp. 147-149.
[22] Ibid., p. 242.

To pursuing excellence in software development, the final product of this process --an executable program and its documentation-- must meet the normal, expected and exciting customers' requirements.

Customer Involvement and Communication

How can we get detailed information about how people work when they cannot articulate it on their own? What is the best way to involve the customer in the design process? Many software designers have been in front of these questions sometime trying to design a first-class product.

Holtzblatt and Beyer[23] addressed these questions through a technique named Contextual Inquiry (CI): "CI allow to get data from customers *in context*: while they work at real task in their workplace". With this technique we can observe, interrupt, ask questions and dialogue with the user at work. Both, the interviewer and the user, can discover beyond the user work and what it is implicit in the user mind. When the user working on or describing its real problems, it is much more expressive than when talking in generalities. According the authors' experience, customers become just another designer among designers, losing its capabilities to represent the user community, when they brought into design meeting.
Entity relationship models (ERM), data flow diagrams (DFD) or other software design tools are unfamiliar to customers, instead, contextual design builds on customers' strengths by doing all work with them in their own context and on their own problem.
The final purpose is to gather data in the user workplace and put the people making design decisions in front the user.

---

[23] Holtzblatt, K. and Beyer, H., "Making Customer-Centered Design Work for Teams", *Communications of ACM*, October 1993, v.36, n10, pp. 93-103.

Name Index

Boudjlida, N. and Bassons, H., "Integration mechanisms in ALF", *Proceedings of the Second International Conference on Systems Integration - ICSI '92*, pp. 315-324.

Coallier, F., "How ISO 90001 fits into The Software World", *IEEE Software*, January 1994, p. 98.

Comaford, C., "Expediting development with Support Teams", *PC Week*, March 15 1993, v10, p. 68.

Constantine, L., "Building structured open teams to work", *Software Development 1991 Proceedings*. Miller Freeman, San Francisco, California, 1991.

DeBrandt, J., "Estimating maintenance costs", *Computerworld*, March 25 1991 v25, p. 69.

Evans, James R. and Lindsay, Williams M., *The Management and Control of Quality* (West Publishing Company, 1993) pp. 150-163.

Evans, James R. and Lindsay, Williams M., *The Management and Control of Quality* (West Publishing Company, 1993) pp. 147-149.

Hauser, John R. and Clausing, Don, "The House of Quality", *Harvard Business Review*, May-June 1988, p. 63.

Holtzblatt, K. and Beyer, H., "Making Customer-Centered Design Work for Teams", *Communications of ACM*, October 1993, v.36, n10, pp. 93-103.

Humphrey, W., *Managing the Software Process*, Addison-Wesley 1989.

Hutchinson, C. and Kihara, T., "QFD as a Software Design Tool for Software Development", *The Fourth Symposium On Quality Function Deployment*, 1992, pp. 1-13.

Krick, K., "Balancing the benchmark opportunities", *IEEE Global Telecommunications Conference and Exhibitions*, 1991, v1, pp. 167-169.

Liston, B. ,"TQM and Software Engineering: A personal Perspective", *The Fourth-Symposium on Quality Function Deployment*, 1992, p. 10.