Total software quality management model

R.J. Soper, B. Karaçali, T.L. Honeycutt
Dept. of Computer Science, North Carolina State University,
Box 8206 Raleigh, NC 27695-8206
E-Mail: tlh@csc.ncsu.edu

Abstract

The Total Software Quality Management Model (TSQM) approaches the development of valid software from the perspective of integrating validation efforts throughout the software development. The model is based on the fundamental premise of aligning software development or refinement with improving a business process that has been directly correlated to the performance of the organization in order to provide measurable return for the organization's software investment. The model approaches the quality of a software product from the users' point of view and assesses the impact in terms of its contribution to improving the business process supported by the software.

Introduction

Software resources constitute a competitive and strategic force in the business operations in the Information Age [1]. Following the significant decreases in the hardware costs, software systems have become an essential part of our daily lives. These trends have resulted in increased application areas, code size and complexity [2] [3].Unfortunately, despite the last couple of decades of experimentation with software development, there are still horror stories. Due to the vital role of software in many critical systems, the production of high quality software has become a major issue. It is argued that one out of every six new large scale systems has been canceled [2]. Approximately half of the software projects exceed their budgets and schedules. and the problem gets worse as the project size increases [2]. It is claimed that "getting the software right the first time is hard even for those who try" [2]. These facts indicate the existence of major problems with software development.

The most costly errors in software development are those that are com-

mitted at the high level design phases [4]. The transition from concept to valid product specifications constitute the most critical component of software development, and thus one of the areas that is prone to result with major problems [5]. Any error, no matter how small it might look at first, may present itself as a serious design flaw later in the development. Consequently, in order to be able to ensure a quality software product, it is important to build the right one in the first place. Problems in understanding the users' expectations and their often informal and vague requirements result in not accurately capturing the information flow of a system and thus implementing a system that does not satisfy the users' information needs. Traditionally, ensuring the validation of a system has been left as one of the items in the testing checklist that is conducted following the completion of the product. Consequently, when major design problems related with validation were encountered major code rewriting was necessary and it was already too costly to correct them.

The problem of achieving valid specifications is further complicated by the fact that user requirements tend to be dynamic. Users do not generally provide very detailed information about their expectations and some level of uncertainty exists in the requirements [6]. It is often the case that the users themselves are not very clear about their information needs [7]. As the uncertainty unfolds some discrepancies in the developers interpretation and the users' expectations may arise and the assumed specifications will have to be re-considered. Handling this inherent change is one of the critical issues in software development. Deriving the right set of specifications that would lead to the implementation of a valid software product constitutes one of the main challenges facing software engineers.

Another major obstacle in producing high quality software, is the problem of accurately measuring the quality of a given software product. It is not possible to improve something that is not clearly perceived. Unlike other engineering disciplines, it is not easy to accurately assess the quality level of a piece of software through employing traditional statistical methods.

The Total Software Quality Management Model (TSQM) provides a resolution to the problems of building a valid product and measuring the quality of a software product by deploying quality control principles. The model views software development from a process based approach. This approach provides a framework for achieving high quality in software products from the users' perspective.

TSQM Model

The fundamental premise of the TSQM model is the alignment of software development with improving a business process. The model is presented in Figure 1.

The model provides a framework for managing Information Technology (IT) on the enterprise level. It is a comprehensive model for IT manage-

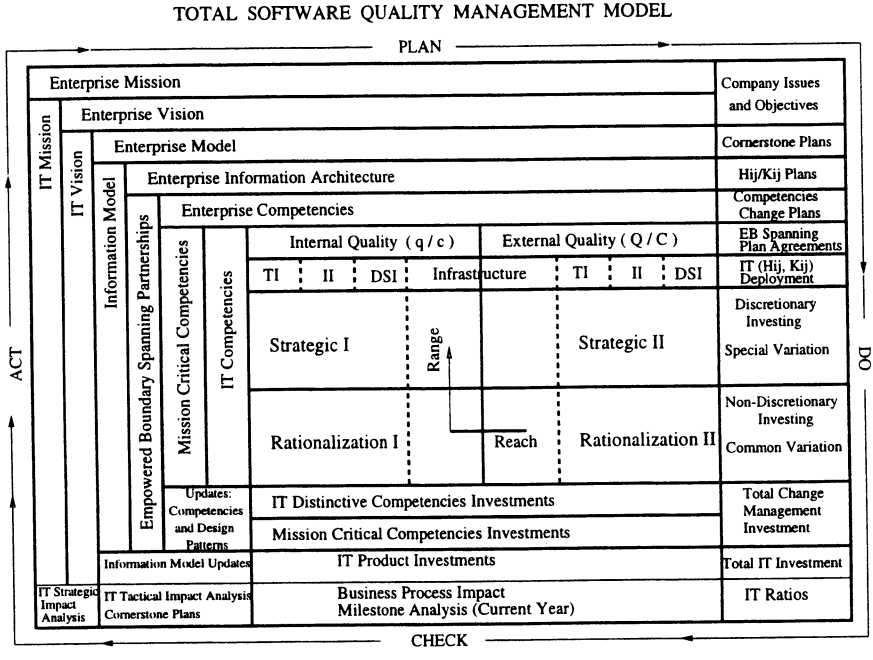

Figure 1: TSQM Model

ment - tightly coupled with a business perspective. The model starts from enterprise issues and moves towards unique software projects that are the core of the information resources of the company.

TSQM is built around a Plan-Do-Check-Act (PDCA) cycle. The PDCA cycle is used by Japanese to improve a business process [8] [9]. The Plan step involves planning the necessary activities based on the current status of a process. The Do step is where the plans are executed. The Check step is intended for checking the impact of the Do step. Based on the analysis conducted at the Check step, the necessary modifications to improve the process are determined. These changes are implemented in the Act step. From this point on the PDCA cycle repeats itself at regular intervals.

At the highest level of abstraction, TSQM has its own PDCA cycle to manage all projects in the enterprise.In addition, this PDCA cycle is applied to every project conducted in the company. The model can be perceived in two levels: enterprise level view and project level view. The enterprise level view involves interactions of the IT department with the overall enterprise with a perspective of managing IT investments. The project level view handles single IT projects in connection with the information architecture of the company. Each level is governed by its PDCA cycle to control the quality of the activities. The project level view is essentially the Do func-

tion of the Enterprise view. The plans determined at the enterprise level are conducted in the project level view.

Enterprise View

The TSQM model's highest abstraction emphasizes the alignment of Enterprise mission and vision with the IT department mission and vision (upper left corner of Figure 1). The enterprise mission involves broad statements about the company's purpose, philosophy and goals. The enterprise vision is a long range statement that matches company's competencies with the environment. This alignment of IT and the enterprise with respect to mission and vision ensures that the decisions regarding the IT resources of the company are consistent and supportive of the business directions for the overall company. The major impact of this alignment is the guarantee that projects implemented are based directly on business needs. The outcomes of this alignment, the company issues and objectives, are depicted on the top right corner of Figure 1.

The enterprise model and information model are aligned at the next level in Figure 1. The enterprise model is a model of the organization and its environment within which the information systems exist. Accordingly, the information model is the description of the information system resources that exists within the organization. The cornerstone plans, are generated based on the enterprise model and the information model.

One of the main issues forming the backbone of TSQM is the enterprise information architecture. It is an umbrella term for the categories of information needed to manage a company's business, the methods the company uses to generate this information, and the rules regulating its flow. This architecture forms the information structure for the company; and hence, every single IT project, mainly software projects, should consider this architecture during requirements identification phases.

The information flow within a company involves various departments. Therefore, the collaboration of these departments on IT projects is an essential requirement for projects to capture the information flow accurately. This requirement is represented in TSQM under the term empowered boundary spanning partnerships which are linked with the enterprise information architecture.

Human resources constitute one of the major IT resources of a company. As the pace of technology is accelerating, the issue of keeping the competencies of the human resources up to date becomes a challenging task for companies. The combination of skills, processes, and tools to deliver the IT mission form the mission critical competencies. While the mission critical competencies have a management oriented nature, the IT competencies are related to the technical expertise necessary to accomplish IT projects. Together they form the enterprise competencies.

The PDCA cycle in the enterprise view is depicted at the outer edges

of Figure 1. The Plan step results in cornerstone plans, Hooshin and Kaizen plans (defined below), competencies change plans, and empowered boundary spanning plans. The Do step involves implementing these plans in accordance with the IT architecture. The total change management investment, the mission critical competencies investment, the IT distinctive competencies investment, the IT product investment, and the total IT investment correspond to the investments incurred during the application of the IT strategic plan.The check part involves analyzing the various IT ratios to measure the effects produced by the IT investments. Total IT Expenditures/Total Revenue denotes the standard IT ratio. Milestone analysis, IT strategic impact analysis, IT tactical impact analysis are conducted to determine the business impact of the IT activities at the Check step. As a result of the check activities, some necessary updates are identified. These modifications to the initial plans and the IT architecture are conducted in the Act step of the PDCA cycle. Updates to the competencies, design patterns, and information model are conducted in the Act cycle.

Project Level View

The project level view assumes a quality approach for all the IT projects that are implemented to support the IT architecture and the business goals as described in the enterprise view. The primary goal of software systems is to support some business function [6]. This goal implies that the software systems should and must provide some discernible improvement to the business function that they are designed to support. TSQM starts with a thorough analysis of the business process under consideration. The model identifies each process as related to an internal quality objective or to an external quality objective as shown in Figure 1. An internal quality objective is related to improving the productivity level and an external quality objective is related to increasing the external customer's perceived quality. The process is further identified based on its state: stable or unstable. The state of the process is determined by its process capability index. The process capability index (Cp) is a statistical variable calculated based on the business process's performance.

Cp=(U - L) / 6s where:
s: sample standard deviation
U: upper limit for customer specification of business process
L: lower limit for customer specification of business process

Cp>4/3 denotes a stable process that undergoes continuous refinement (Kaizen) [10]. This continuous improvement involves identifying the root causes of common variations due to hardware and software. Since the process is stable, these and other variances (i.e. work procedures) are independent and add up to the total variance. Therefore the individual variances can be isolated and reduced individually. Identification of these root causes help in determining the new requirements for the software product that will

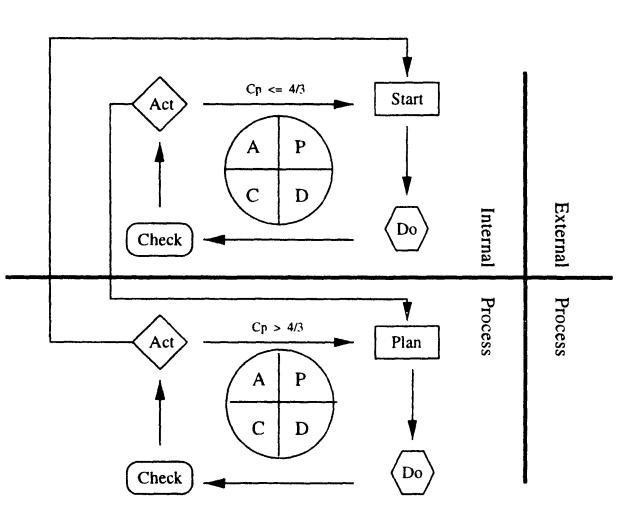

Figure 2: Procedure for Improving a Process.

support the business function. The analysis of the business function clearly defines the specifications for the refined software product.

Cp$\leq$4/3 denotes an unstable process that suggests major innovative changes in procedures (software) and equipment (hardware). As in the case of a stable process, the root causes of the special variations are identified. This unstable process undergoes a breakthrough analysis (Hooshin) [11] to further identify feasible, cost efficient solutions for improvement. Process simulation models are one of the major tools utilized in conducting Hooshin Analysis. In this case the business process is not merely automated but it is virtually reengineered [12]. The results of the Hooshin Analysis, produces clear functional requirements as outputs, for the software that will support this unstable process. The results of the Hooshin Analysis will provide the sufficient information for the identification of valid requirements.

The 8 step procedure [9] for improving a process that is based on capability index in shown in Figure2. The first step is the Start step on the upper right corner of Figure 2. The existing plans are implemented at the Do step. The checking of the impact is conducted at the Check step. When the Act step is reached a decision is made. If the process is unstable (Cp$\leq$4/3) the process stays in the upper PDCA cycle. However if it has become stable, (Cp>4/3) then it moves to the plan node of the lower part of the diagram and follows the PDCA cycle for continuous refinement. If the process eventually become unstable, this is captured at the Act step and a decision to move to the upper PDCA or to stay in the lower PDCA cycle is made based on the Cp value. If Cp>4/3, process stays in the continuous refinement cycle, else it moves to the reengineering cycle.

This 8 step procedure is applied to every software project in the IT department. As expressed before, the project level view is essentially the Do step of the higher level enterprise View. The project level view is depicted in Figure 3, which is a detailed version of the core part of Figure 1.

The 8 step procedure pattern can be recognized in Figure 3. In the PDCA cycles of each quadrant, the following steps are executed. TSQM starts with the analysis of the business process, regardless of its state, as the first step. The second step, after the specifications are correctly translated into product specifications is implementing the software that will provide the identified functionalities. The third step involves checking that the implementations are conducted correctly and analyzing the impact of the software product on the process. In the fourth step, impact analysis is conducted and necessary improvements to the initial set of specifications are determined. The model then starts looping by going to the first step and repeating the cycle. It is important to note that the cycle is Check-Act-Plan-Do to initialize the model.

The TSQM model encompasses the overall organization by applying the process described above to every project in the company. The model requires all new software development or refinement projects to fall into one of the four investment management quadrants: Strategic I, Strategic II, Rationalization I, Rationalization II as depicted in Figure 1. Processes in quadrant Strategic I are related with the internal customer for the purposes of productivity improvement and require a major reengineering/innovation (Hooshin). Processes in Strategic II are related with increasing the customer's perceived quality and require Hooshin Analysis. The quadrants Rationalization I and II include projects that are associated with processes undergoing continuous refinement in productivity increase or external quality increase, respectively. Each individual project can be in any one of the above quadrants based solely on the capability index of the process that is being implemented. The variations in the processes related to projects in the Rationalization I and II quadrants have lower variations and they are more stable. The transitions between the quadrants are depicted in Figure 3 in more detail.

As the process becomes more stable it moves to the PDCA cycle governing the stable processes that are undergoing rationalization. If the process becomes unstable, it moves to the strategic quadrants where unstable processes are looping around the PDCA cycle of their own.

The outcomes of stable processes, which have reduced variation, are non-discretionary investing. The outcomes of the unstable processes are discretionary investing. Hij/Kij contract deployment refer to the hooshin and kaizen investments related to business unit i and business objective j, respectively.

The TSQM model ensures the validity of the software by deriving the specifications of the software product, based on the root problems that

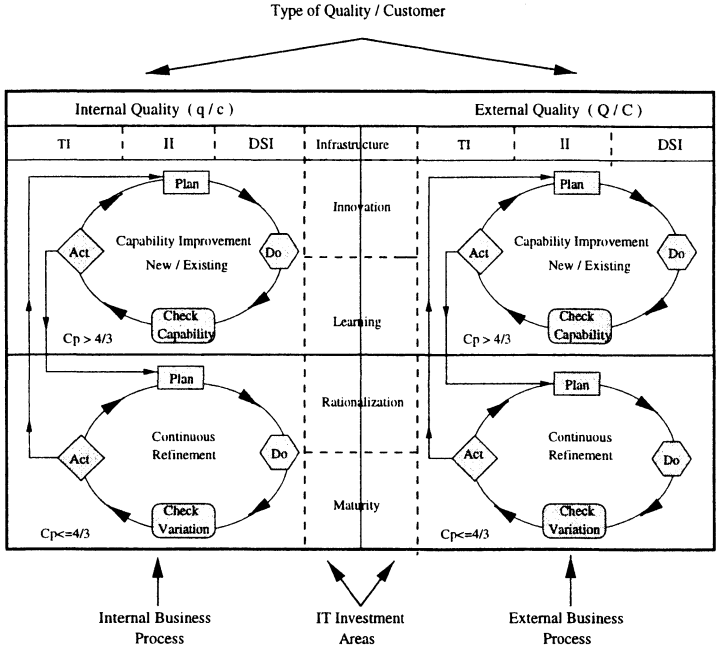

Figure 3: Do Function.

constitute the variance for the business process. The model guarantees that the functionality provided by the software does in fact correspond to the needs of the business process. Starting from the business process, the model captures the process flow and integrates it into the software product. The information architecture of the overall organization plays a critical role in the analysis of the process needs. In this way product validation functions as the core engine for software development process. Hence, the model addresses the problem of building a valid product.

The PDCA cycle provides an iterative approach to developing the right set of specifications. The apparent need for a mechanism to handle the dynamics of user requirements is addressed by the model which integrates continuous change management into development. The PDCA cycle will be repeated until the right set of specifications are achieved.

The TSQM model also addresses the problem of measuring the quality of a piece of software. There are many views of quality in the literature, including transcendental view, user view, manufacturing view, product view and value based view [13]. Transcendental view considers quality as something that can be identified rather than clearly defined. User view treats quality as "fitness for use". Manufacturing view considers quality as "conformance to specifications". The product view ties quality "to the inherent characteristics of the product". Value based view treats quality as the

amount on which the customers are willing to pay for it. Kotler defines quality as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [14]. This definition and the user view emphasize the importance of the ability of the software product to satisfy the needs of the users, in the case of software products, the information needs of the end users. The TSQM model emphasizes the software system's ability to improve a business process, and thus meeting the needs of the users and producing an discernible impact on the overall business.

The TSQM model tries to measure the quality level of a software product indirectly by measuring its impact on the outputs of the business process. If a process prior to the implementation of a system has capability C, and has a capability level C' after the implementation of the system, the model argues that the software system accounts for the difference in the capabilities. Depending on the level of improvement incurred and assessed by the capabilities, the quality of the software can be determined.

Conclusion

TSQM approaches the building of a valid product from the perspective of integrating validation efforts throughout the software development. The model treats software products as utilities with the sole purpose of improving a particular aspect of a business process. The requirements of the business process determine the specifications of the software that is to be built. Starting from this view point, any functionality required for improving the business process is implemented in the software product. This approach guarantees that the software that is built meets the needs of the users and the necessities of the business, otherwise it would not have been built in the first place.

In addition to building validation into software development, the TSQM model proposes a quality assessment method to software products. Assuming a user view to quality, the model determines the quality level of the software product based on its ability to improve the business process that it was supporting. The outputs of a business process can be accurately measured and the quality level of the process at any point in time can be assessed by its variance with respect to the low and high limits. The model, by comparing the calculated capability indexes for a process prior to and after the implementation of the software system, can determine the perceived quality of the software product. TSQM offers a company wide quality control to the implementation of the enterprise information architecture by applying this methodology to every project in the organization.

Keywords

Software Engineering, Total Quality Management in software development, Information Technology Management, Measuring software quality.

References

[1] C. Jones. Globalization of software supply and demand. *IEEE Software*, pages 17–24, Nov. 1994.

[2] W.W. Gibbs. Software's Chronic Crisis. *Scientific American*, pages 86–95, Sept. 1994.

[3] A.K. Onoma. Practical Steps Toward Quality Development. *IEEE Software*, 12(5):68–76, Sept. 1995.

[4] R. Guindon. The process of Knowledge Discovery in System Design. In *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, Amsterdam, 1989. Elsevier Science Publishers B.V.

[5] R.W. Zmud T.A. Byryd, C. Cossick. A synthesis of Research on Requirements Analysis and Knowledge Acquisition Techniques. *MIS Quarterly*, pages 117–137, Mar. 1992.

[6] R.S. Presman. *Software Engineering: A practitioner's Approach.* New York, NY:McGraw-Hill Companies, 1997.

[7] J.F. Rockart. Chief Executives Define Their Own Data Needs. *Harvard Business Review*, pages 81–93, March-April 1979.

[8] S. Mizuno. *Company-Wide Total Quality Control.* Tokyo, Japan: Asian Productivity Organization, 1988.

[9] Edelstone S. E. H. V. Roberts. Practicing TQ to Learn TQ. In *TQM University Challenge*, Spartangburg, SC, May 10-16 1992. Milliken Company Corporation.

[10] M. Imai. *Kaizen.* New York, NY: McGraw-Hill Publishing Company, 1986.

[11] B. King. *Hooshin Planning: The Development Approach.* Methuen, MA: Goal/QPC, 1989.

[12] P.F. Drucker. The coming of the New Organization. *Harvard Business Review*, January-February 1988.

[13] S.H. Pfleeger B.Kitchenham. Software Quality: The Elusive Target. *IEEE Software*, 13(3):12–21, Jan. 1996.

[14] P. Kotler. *Marketing Management: Analysis, Planning, Implementation, and Control.* Englewood Cliffs, NJ: Prentice-Hall Inc, 1994.