Review

Recent Advances in Software Quality Management: A Review

Boby John¹, R. S. Kadadevaramath² and Edinbarough A. Immanuel³

Abstract

Many organizations has been utilizing the benefits of information technology to gain competitive advantage in their respective businesses. As a result, the number of software development companies increased many folds during the last three decades. As the competition increased, the need for delivering good quality software within the committed schedule also increased. Even today many software companies have to deal with the consequences of delivering poor quality products, schedule and cost overrun problems. Many software quality and development frameworks have been suggested in the past to get rid of the aforementioned problems and a lot of research has been carried out in the field of software quality management and practices. In this paper, the authors provide a review of the major research works published in the field of software quality management. The study found that the research works in the software quality can be classified into five categories namely studies exploring the relationship between software quality and (i) total quality management implementation, (ii) adoption of quality management systems like ISO 9000 series of standards, (iii) capability maturity model level, (iv) development of defect prediction models and (v) quality management and evaluation practices of specific categories of software development process. The authors also tried to identify the gaps in existing research and future areas of research in software quality management.

Keywords: Capability maturity model, Defect prediction models, ISO 9000, Quantitative project management, Software quality, Total quality management

INTRODUCTION

Surviving in a globalised business world is not easy unless the organizations have a competitive advantage (Kanji and Asher, 1999; Samason and Terziorski, 1999 and Adam et al., 2001). During the past three decades, many organizations deploy information technology (IT) to gain the aforementioned competitive advantage. The IT industry has witnessed tremendous growth in the last three decades. The Gartner group estimated even a decade ago that the worldwide user spending on software exceeded $730 billion and that in the packaged software market topped $176 billion (Shiffler, 2003; IDC, 2003). Many business organizations are using IT as a means to achieve operational efficiency, improved productivity and service quality and responsiveness (Mooney et al., 1996).

As the demand for software products increased, the number of software development firms also increased. Even though the number of software development firms increased rapidly, delivering quality software products without cost and schedule overrun has been always a challenge in software industry (Pearson et al., 1995; Phan et al., 1995). A study by US General Accounting Office (1973) reported that many government software projects were never delivered or couldn’t be used and had cost over runs or schedule overruns. The breakup of software project problems (Osmundson et al., 2002)
published by the study is given in figure 1. Many software firms has comprised on quality to meet the schedule pressure and control the development costs leading to detecting higher number of defects at customer end (Kemerer, 1997). The lack of quality had significant costs to the suppliers who face dissatisfied customers, loss of market share and rework of rejected systems (Sibisi and Waveren, 2007). Hence producing quality software within the committed time and cost are very important for the survival of software development organizations in the highly competitive information technology industry.

Defining software quality is not easy and there is no single adequate measure for software quality. Fenton and Bieman (2014) suggested measuring software quality in terms of delivered defect density. The delivered defect density is the number of defects per unit size. According to ISO/IEC 9126 – 1, the software quality model has six characteristics namely functionality, reliability, usability, efficiency, maintainability and portability (Al-Kildar et al., 2005). ISO 9126 (1991) defined software quality as the totality of features and characteristics of a software product that bear on its ability to satisfy stated and implied needs of the customer. The widely adopted Capability Maturity Model (CMM) developed by Software Engineering Institute (SEI) of the Carnegie Mellon University classifies the software process into five maturity levels namely initial, repeatable, defined, managed and optimizing (Paulik et al., 1994). In short, there has been a variety of approaches and guidelines suggested for software quality management and improvement.

The purpose of this paper is to review the literature on recent advances in software quality management and identify the major gaps in the research. The remaining part of the paper is organized as follows: The review methodology is described in section 2, the literature review analysis is given in section 3 and the conclusions and areas of future research are summarised in section 4.

REVIEW METHODOLOGY

In the recent past numerous books have been written, papers have been published, conferences have been organised on various approaches for measuring and improving software quality and software quality management practices. The aim of this paper is to provide a comprehensive review on the recently published scientific literature on software quality management and related topics.

The review process started with searching for relevant articles. The focus is on papers and books published in English from 1990 onwards. Only articles published in journals, books or presented in conferences are included in the review. The authors searched the popular research databases like ScienceDirect, ieee xplor, google scholar and researchgate using software quality, software quality management or software quality prediction as key words.

Six hundred and twenty four articles suggested by the databases are identified for the survey. After going through the key words and abstracts, 196 articles were shortlisted for detailed study. Another 79 papers were also excluded after reading the full text as they were not directly related to the topics of interest. The remaining 117 articles were reviewed in this study. The details are given in figure 2.
Table 1. Category wise number of articles reviewed

<table>
<thead>
<tr>
<th>Number</th>
<th>Category</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total Quality Management</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Quality Management Systems</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Capability Maturity Model</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Quality Prediction Models</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>Others</td>
<td>13</td>
</tr>
</tbody>
</table>

The details of review analysis for each groups is as follows:

**Literature Review Analysis**

The authors carried out detailed analysis on the articles selected for the study and classified the articles into five different categories based on the topics of their research. The grouping had enabled the authors to summarise the findings and identify the future research areas. The number of papers reviewed under each category is given in table 1.

**Total Quality Management and Software Quality**

The universal acceptance of the TQM has inspired software firms to adopt it to their industry (Cortada, 1995, Alkhafaji et al., 1998; Bhattacharya et al., 1998). Parzinger and Nath (2000) studied the relationship between TQM implementation factors and various measures of software quality. The study showed that the TQM implementation factors have significant positive correlation (p value < 0.01) with the software quality measures except cost of quality (p value > 0.05).

Li et al., (2000) suggested an approach to instil TQM method in software development process. Issac et al., (2004) proposed a holistic conceptual framework for implementation of TQM in software industry. The framework suggest to constantly measure the performance of the system using metrics, analyse with respect to the benchmarks set and provide feedbacks to the system to take necessary corrective steps. Carrol (1995; Camuoff et al., 1990) argued that the application of the key elements of TQM in software development has the potential to improve the software quality. Many researchers (Munson and Khoshgoftaar, 1992; Zardony and Tumanic, 1992; Gong et al., 1998) studied the application of TQM concepts like statistical quality control and quality function deployment on software quality. Walrad and Moss (1993) observed that the impact of TQM techniques on system quality depends on effective linking of product and process metrics to system quality objectives.

Even though the research suggested positive correlation between TQM implementation and software quality, most of the works were on conceptual framework or empirical studies based on the analysis of survey data.

**Quality Management Systems and Software Quality**

The International Organization for Standardization (ISO)

Shem et al, (2015) showed that generally there exists positive correlation between implementation of standards and product quality. Jung et al, (2004) conducted a survey among users of a packaged software product to evaluate the structure of the software quality model proposed in ISO /IEC 9126 – 1. The study revealed that ambiguities exist in the way that ISO/IEC 9126 is structured and the sub characteristics categorization isn’t consistent with the ISO/IEC 9126 definition. Balla et al., (2001) published a case study on the success of ISO certification in a software company in Hungary. According to the study, working in conformance with QMS caused about 30% of extra effort in small projects and about 10 – 20 % in bigger projects. But the advantages were perceived to outweigh the costs. Yang (2001) studied the attributes of the software quality given in ISO 9000 standard and its usefulness in estimating the software product quality. Paulk (1993) observed that there exists strong correlation between ISO 9000 standard and CMM frame work. Franch and Carvallo (2003) developed a quality model for choosing off the shelf software packages based on ISO / IEC 9126 – 1 standard. Coallier (1994) explored the fitment of ISO 9000 standards for software development process. The study found that ISO 9001 / 9000 – 3 only partially supports the assets needed to deliver the software when it is required with minimum life cycle costs. Al – Kilidar et al., (2005) reported an empirical study on application of ISO / IEC 9126 to software design documents and found that the standard has ambiguity and overlap in some concept definitions, doesn’t consider reliability and maintainability as well as validity and modularity of design products. Jenner (1995) showed how ISO 9001 can be used for software development process.

The review of the literature had thrown out conflicting findings. While many of the studies suggested improvement in quality with the implementation of quality management systems or standards, there were studies disagreeing with the aforementioned conclusion. Unfortunately most of the papers were empirical studies based on survey data.

**Capability Maturity Model and Software Quality**

The capability maturity model (CMM) developed by the software engineering institute (SEI) of Carnegie Mellon University is one of the widely accepted framework for characterising the capability of software development processes (Pressman, 2005).

Harter et al., (2000) empirically investigated the relationship between process maturity measured in CMM maturity scale, product quality, development cycle time and effort. The study found that 1% improvement in process maturity is associated with a 1.589% increase in product quality. The other findings of the study are higher the quality, lower is the cycle time and development effort. Herbsleb et al., (1997) also showed that CMM levels influence the software quality and project performance variables. Zimmerman (2001) suggested that poor planning and lack of training are two root causes of software project failure. Many other studies also showed that higher CMM levels are associated with improved software quality (Zimmerman, 2001; Lawlis et al., 1995; Krishnan, 1996; Subramanian et al., 2007).

But Hansen et al., (2004) pointed out that the evidence for higher CMM levels are associated with higher quality are limited and not based on reflective models.

Higher maturity levels (level 4 and 5) of capability maturity model requires the quantitative management of the process (C P Team, 2006), which means the process need to be controlled by statistical and other quantitative techniques. Tamura (2009) developed three process performance models using regression techniques, one for achieving product quality objectives (defect density) by controlling requirement inspection rate (pages per hour) and prototype developed or not. The second one for achieving code review yield (% of defects present in the software that are removed by the review) targets by controlling review rate (the number of lines of code reviewed per hour). The third model is to manage the escaped unit test defect density using test coverage as controllable variable. Hao and Zhang (2011) also developed a model for delivered defect density in terms of average team skill level and test coverage as controllable variables.

Majority of the papers suggested that higher maturity levels were attached with better software quality. Except the research on developing process performance models, the remaining articles were mostly based on empirical evidence.

**Development of Defect Prediction Models**

From 1970s onwards one major area of research in software quality has been the development of defect prediction or classification models. The important papers published on defect prediction models from 1990s onwards along with the methodology used is summarised in table 2.

The breakup of different techniques used to develop the prediction models is given in figure 3. Most of the models used static code attributes like code complexity, etc as predictors those are not routinely measured by the project managers and may not be under their control.
Table 2. Research on developing defect prediction models

<table>
<thead>
<tr>
<th>SL No</th>
<th>Methodology</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Regression Analysis</td>
<td>Regression with different intercepts and slopes (Nagappan et al., 2005), Logistic regression (Khoshgoftaar and Allen 1999; Cruz and Ochimizu 2009), Zero-inflated negative binomial regression (Succi et al., 2001), Zero – inflated Poisson regression (Khoshgoftaar, 2001), Regression via classification (Bibi et al., 2006), Linear mixed-effects regression models (Binkley, 2007)</td>
</tr>
<tr>
<td>2.</td>
<td>Multivariate Analysis</td>
<td>Factor analysis (Khoshgoftaar and Munson, 1990; Munson and Khoshgoftaar, 1990), Principal component analysis (Niel, 1992; John and Kadadvevaramath, 2014), Discriminant analysis (Schneidewind, 2001; Khoshgoftaar and Seliya, 2002), Cluster Analysis (Sandhu et al., 2010), Singular value decomposition (Sheriff et al., 2007)</td>
</tr>
<tr>
<td>3.</td>
<td>Machine learning techniques</td>
<td>Neural Networks (Khoshgoftaar et al., 1995; Khoshgoftaar et al., 1997; Mair et al., 2000; Kanmani et al., 2004; Pedberg et al., 2004; Thwin, and Quah, 2005; Bezerra et al., 2007; Kanmani et al., 2007; Singh et al., 2008; Tao and Wei-hua, 2010), Bayesian Networks (Fenton and Niel, 1999; Amazaki et al., 2003; Fenton et al., 2007; Menzies et al., 2007; Pai and Dugan, 2007; Turhan and Bener, 2007; John, 2012), Classification and Regression Tree (Porter and Selby, 1990; Khoshgoftaar et al., 1999; Kur and Liu, 2005; Menzies et al., 2003; Khoshgoftaar and Seliya, 2003b; Khoshgoftaar and Seliya, 2002; Knab et al., 2006; Ceylan et al., 2006), Random Forest (Guo et al., 2004; Kaur and Malhotra, 2008), Rough Sets (Morasca and Ruh, 1996; Yang and Li, 2008), Instance based learning (Ganesan et al., 2000; Emam et al., 2001; Khoshgoftaar and Seliya, 2003a; Khoshgoftaar et al., 2006; Challagulla et al., 2006), Support Vector Machines (Elsh and Elsh, 2008), Self Organizing Maps (Reformat et al., 2003; Mahaweerawat et al., 2007), Genetic Programming (Afzal and Torkar, 2008; Afzal et al., 2008), Particle Swarm Optimization (De Carvalho et al., 2008; Cong et al., 2010; De Carvalho et al., 2010), Grey Prediction Theory (Zhu and Wu, 2009), Dempster-Shafer Belief Networks (Guo et al., 2003), Fuzzy Logic (Xu et al., 2000; Yuan et al., 2000; Reformat, 2003; Yang et al., 2007; Hribar and Duka, 2010), Recency Weighting Technique (Joshi et al., 2007), Spam Filter Approach (Mizuno et al., 2007), Weighted Similarity Modelling (Nagwani and Verma, 2010), Association Rule Mining (Song et al., 2006)</td>
</tr>
</tbody>
</table>

Moreover software quality depends on people related factors like programmer skills, level of expertise, domain knowledge, etc (Antony and Ferguson, 2004). Hence the process performance models developed as per the guidelines of CMM frame work would be more useful for quantitatively managing the software development process.
### Table 3. Management and evaluation practices of specific categories of software development

<table>
<thead>
<tr>
<th>SL No.</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Comparison of quality of the software developed by distributed teams and collocated teams (Bird et al., 2009)</td>
</tr>
<tr>
<td>2</td>
<td>Development of metric based approach and software engineering metrics to ensure the quality of systems developed using commercial off the shelf components (COTS) (Sedigh-Ali et al., 2001; Rawashdeh and Mataalkah, 2006)</td>
</tr>
<tr>
<td>3</td>
<td>Comparison of quality of software as a service (offering software using a subscription model) with software offered using perpetual licensing model (Choudhary, 2007)</td>
</tr>
<tr>
<td>4</td>
<td>Comparison of quality of open source applications with that of commercially developed software (Stamelos et al., 2002; Raymond, 2001; Aberdour, 2007)</td>
</tr>
<tr>
<td>5</td>
<td>Application of quality function deployment (QFD) on improving software quality (Liu, 2000; Islam and Hasin, 2014)</td>
</tr>
<tr>
<td>6</td>
<td>Application of fuzzy logic on quality improvement (Liu et al., 2006), effect of project management policies on software quality (Garcia et al., 2008), effect of various software process improvement (SPI) methodologies on software quality (Ashrafi, 2003)</td>
</tr>
</tbody>
</table>

### Management and evaluation practices of specific categories of software development process

Apart from studying the effect of TQM, ISO, CMM and defect prediction modelling on software quality, a lot of research has also been carried out in the field of management and evaluation practices of specific categories of software development process. The important among them are summarized in Table 3.

### CONCLUSION AND FUTURE RESEARCH WORK

The paper presented a literature review on software quality management practices. The review revealed that the important articles published recently can be classified into five groups namely those studying the impact of (i) TQM implementation on software quality, (ii) ISO 9000 certification on software quality, (iii) adoption of CMM framework on software quality, (iv) development of software quality prediction models and (v) Management and evaluation practices of specific categories of software development process. The published articles suggested that there is positive correlation between TQM implementation and software quality as well as CMM framework and software quality. These studies are carried out using empirical data collected through survey. The future researchers can fine tune these findings through correlating quantitative measures of software quality with that of TQM, ISO, CMM implementation. The researchers can also establish the correct relationship between ISO certification and software quality as the past studies showed that ISO certification has conflicting results.

A lot of research has been carried out on developing software quality prediction models to either predict the software faults or classify software modules as fault prone or not. Some of the drawbacks of these models are as follows: There is no one best model suitable for all types of software development. In fact the research on comparison of these models suggested that different models need to use for different type of scenarios. Moreover majority of these models use static code attributes like code complexity, etc as predictors. Many of these factors are not routinely measured by the project managers and also many of these factors may not be under the control of project managers. Most of these models used data available in the public domain like NASA website, etc. The research on developing fault prediction models using industry data are still in the nascent stage. The review showed that process performance models developed under the quantitative project management process area of CMM framework are more suitable for quantitatively manage the software development process. But these models are mostly based on regression or simulation techniques. The development of process performance models using different algorithms, especially the machine learning algorithms will be an important area of future research.

Moreover the review showed that there is good correlation exit between software quality, productivity, cycle time, and development effort. Hence it is required to develop models or methodologies to quantitatively manage multiple performance characteristics simultaneously. Moreover many of the performance characteristics are qualitative. Hence the models for simultaneously monitoring of quantitative and qualitative performance characteristics will be another important area of future research.

### REFERENCES


Afzal W and Torkar R (2008). A comparative evaluation of using genetic...


Raymond ES (2001). The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary, O'Reilly Media, Inc.