Information & Management 47 (2010) 372–379

Contents lists available at ScienceDirect

Information & Management

journal homepage: www.elsevier.com/locate/im

Total quality in software development: An empirical study of quality drivers and benefits in Indian software projects

Marcus A. Rothenberger [a,*], Yi-Ching Kao [b], Luk N. Van Wassenhove [c]

[a] Department of MIS, University of Nevada Las Vegas, 4505 Maryland Pkwy, Las Vegas, NV 89154-6034, USA
[b] Menlo College, Atherton, CA, USA
[c] INSEAD, Fontainebleau, France

ARTICLE INFO

Article history:
Received 24 April 2009
Received in revised form 8 February 2010
Accepted 23 July 2010
Available online 13 October 2010

Keywords:
Software quality
Total quality management
Quality control
Software Productivity
Error rate
Empirical study of software projects

ABSTRACT

Building upon the software quality and productivity literature, we proposed a construct of *development quality* as the key determinant of *software development productivity* and *product quality*. We validated the model by analyzing software project data collected from a benchmarking consortium in India. Our empirical results showed that an increase in development quality was positively associated with increases in both, development productivity and product quality, while we controlled for the impact of other productivity and quality factors. Our work highlighted the importance of concentrating on quality efforts during the development process, which is consistent with the use of Total Quality principles in manufacturing.

1. Introduction

Software development is similar to other engineering disciplines: a managed approach is likely to improve the quality of the product; thus manufacturing-based concepts have been introduced into the software development process. Several authors have suggested that quality programs based on a TQM philosophy may be able to address the need for quality improvement in software development [18]. One of its major points was that quality could be achieved by incorporating TQM into the production process, rather than by post-process inspection and testing [20]. The CMM and ISO 9000 standard were both based on TQM principles. Thus one of the central aims of CMM and its update (CMMI) involved quality improvement through a reduction of the number of defects by focusing on improvements to the development process.

The adoption of a quality program requires a substantial investment for an organization. Therefore, it is important to evaluate its ROI. Other than fulfilling its original objective "to enhance product quality", it is important to know how the quality development process can affect software productivity. Many factors can affect software development productivity, such as tool usage, development methodology, complexity, and size; however, there has been relatively little literature discussing the role of software quality in this context.

Contradicting views of the quality–productivity relationship have been discussed [8]. On the one hand, many developers believe that there is a trade-off between productivity and quality, since quality enhancement requires additional effort, it may reduce productivity. On the other hand, quality experts assert that quality is a driver of productivity, because reducing the number of product defects can decrease the amount of rework, thereby enhancing productivity. There is anecdotal evidence that some software developers employing process-centered quality practices experience improved productivity. Existing empirical models on the quality–productivity relationship in software development have not investigated the effects of the quality-level obtained during code development on post-release product quality and development productivity, which is central to TQM-based efforts. Thus, our central research question was:

> What are the effects a higher-quality development process on software product quality and software development productivity?

To address this, we developed and empirically validated a model that incorporated TQM philosophy into the quality–productivity relationship of software projects. Earlier studies had used a single quality construct that measured the quality of the final software product [15] or that of the development process

* Corresponding author. Tel.: +1 702 518 5518; fax: +1 702 895 0802.
E-mail address: marcus.rothenberger@unlv.edu (M.A. Rothenberger).

0378-7206/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.im.2010.10.001

[18,19]. We considered both *development quality* and *product quality* in our model and examined the effects of improving development quality on software product quality. *Development quality* captures the software quality during the development process (before release), while *product quality* represents the software quality after release. Consistent with TQM, we placed *development quality* at the center of the *quality–productivity* relationship. In addition, our research investigated the impact of specific aspects and practices on the quality and productivity measures.

We therefore hoped to confirm that a higher-quality development process would yield higher-quality software products and then examine whether an increase in software development quality lead to an increase in development productivity. If development quality improvements positively impacted productivity, the cost of adopting a quality program may be assessed by subtracting the productivity gain. Finally, to provide managerial implications for software management, we identified the process factors that drive development quality. We also examined factors other than development quality that may impact product quality and development productivity levels.

Our model was empirically validated using a database of 60 business-related software projects derived from a benchmarking consortium organized by the Quality Assurance Institute (QAI) India that includes a number of leading Indian software firms. The QAI database is suitable for several reasons:

- The Indian software industry has risen in prominence and importance over the last 15 years;
- Many Indian software firms are noted for their excellence in software development; and
- The QAI India database includes project information on pre-release and post-release defects, allowing for the measurements of both *development quality* and *product quality*.

As a result, the critical role of software development quality could be explored in our study.

2. Theoretical foundation

2.1. Total quality in software development

Inspired by the TQM principles of process improvement, the CMMI has emerged as a roadmap to evaluate and facilitate the software development process. The Software Engineering Institute's (SEI) CMMI and its predecessor, the Capability Maturity Model (CMM) were developed to help the US Department of Defense (DoD) to assess the ability of their software development contractors. Later, the SEI was instrumental in facilitating a broad participation in the model's further development; due to this initiative, CMM and CMMI are now being used by many software development organizations in the US and by outsourcing contractors worldwide [12].

Nevertheless, many software developers initially encountered difficulties in incorporating the quality paradigm into their production processes. In contrast to physical goods, software has no physical appearance, is relatively volatile, and has fewer existing quality metrics. Therefore, the software quality paradigm had to incorporate a complete model of all aspects, including value, attitude and methodology [9]. A number of studies have reported evidence, mostly drawn upon the cases of U.S. companies, on payoffs of software quality improvement efforts [19].

2.2. Prior research in software productivity and quality

After 1990, the focus of research on TQM in software production gradually shifted towards the impact of software quality on software process improvement. The number of organizations implementing the CMM-based software process, primarily due to the DoD's insistence on its evaluation in order to bid on contracts, grew substantially. Studying organizations that engaged in CMM-based activities, researchers showed that software process improvement activities could lead to less operating cost and better product reliability [14]. However, these benefits did not necessarily emerge together since there were trade-offs among them. MacCormack et al.'s [16] study discussed the trade-offs between the productivity and quality benefits and found that only one practice, early prototype during development, was associated with both higher productivity and higher product quality.

A number of empirical studies have addressed the quality and other impact of software process maturity based on CMM. Krishnan et al. showed that implementing CMM significantly increased software product quality (based on post-release defects) and life-cycle productivity (both in development and maintenance); moreover, projects with higher product quality were associated with higher life-cycle productivity. Harter et al. showed that there were direct and indirect effects (due to changes in development quality) when there was an increase in the CMM process maturity level. They found that higher levels resulted not only in higher quality (based on pre-release defects) but also in increased cycle time and development effort. Nevertheless, the increasing development quality level significantly reduced the cycle time and the development effort by more than that of the increase due to the higher maturity level. Consequently, the net effects of process maturity were decreased cycle time and development effort. Analyzing a longitudinal data set over a 10-year period, Harter and Slaughter found that higher process maturity level was associated with higher development quality, which then led to lower IT infrastructure cost.

Some of these studies examined software defect rate prior to product release, development quality, and others focused on the software defect rate after product release, product quality. However, none explored how development quality can impact product quality. Nor did they investigate the effect of development quality on development productivity.

In Table 1, we summarized the categories of productivity and quality factors addressed by the studies (including those prior to the turn of the century). It should be noted that most of these studies demonstrated a nonlinear relationship between the explanatory factors and software quality and productivity. We therefore included all the factors.

3. Research model

The relationships proposed in our model are illustrated in Fig. 1.

3.1. Research hypotheses

Development environments operate with different degrees of quality. In some, code is being developed with few errors; in others, initially error-prone software is improved by testing. It seems very likely that software with a small number of errors in its initial development will contain less after its release. Thus, higher quality development process will ultimately lead to a higher software product quality. The idea is consistent with the quality approach. In an empirical study on the effects of the CMM and ISO 9000 adoption on software development performance, Issac et al. [13] found that these quality certification helped the operational performance of the software firm and helped the organization develop improved products. We therefore proposed quality methodology independent hypotheses, as follows:

Table 1
Software productivity and quality factors identified in the literature.

Factor category	Walston and Fellix [22]	Bailey and Basili [4]	Vosburgh et al. [21]	Boehm [7]	Maxwell et al. [17]	Krishnan et al. [15]	Harter et al. [10]	Harter and Slaughter [11]	Agrawal and Chari [1]	Corresponding factor names in our research
Requirements gathering, analysis and design activities	✓	✓	✓			✓				Design resources (*DESIGN*)
Process maturity, development methodology	✓		✓	✓	✓	✓	✓	✓		Development standard (*STANDARD*)
Software complexity	✓	✓	✓	✓			✓			Project complexity (*COMPLEX*)
Personnel capability		✓	✓	✓		✓				Personnel capability (*PERCAP*)
Project Size			✓	✓	✓	✓	✓	✓	✓	Project size (*SIZE*)
Tool and technology usage			✓	✓	✓	✓		✓		Tool capability (*TOOLCAP*)

H1: Higher development quality leads to higher product quality.
H2: Higher development quality leads to higher development productivity.

3.2. The quality and productivity factors

3.2.1. Development standard

CMM and CMMI emphasize the concept of standardization. They provide a set of engineering and management processes for software production. such as a set of project management controls (requirements management, software project planning, tracking and oversight, software quality assurance, and software configuration management). The assumption is that adherence to these standards will result in a better software development process and increased software quality [2].

3.2.2. Tool capability

Many software developers use tools to improve their development process. Such tools support diagramming, programming, quality assurance, and database conversion. When they are deployed appropriately, there is consistency between the project design and the code, reduced coding errors, and therefore enhanced product quality; their use can directly increase the productivity of software engineering staff.

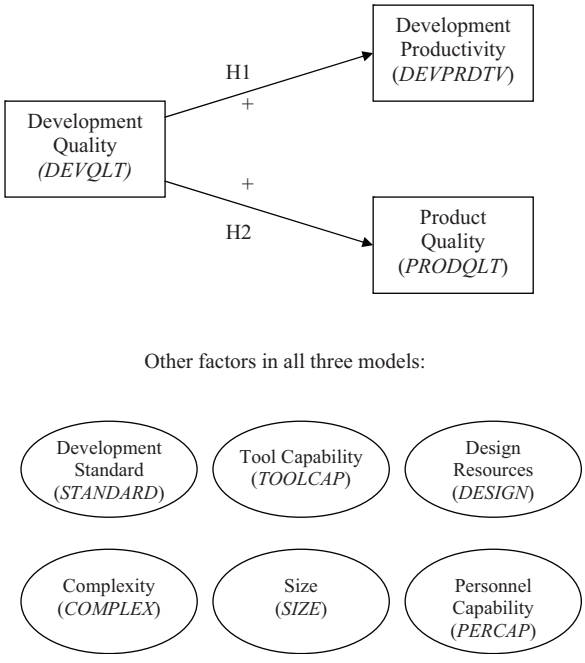

Fig. 1. The relationships between Development Quality, Product Quality and Development Productivity.

3.2.3. Design resources

Well-designed software can be better integrated into the overall system landscape. In addition, the product is more likely to meet customer requirements. The proportion of resources dedicated to pre-implementation activities determines *what* will be delivered. Krishnan et al. showed that greater resources allocated for project design in the early stage of the life cycle can result in higher quality of the software delivered.

3.2.4. Project complexity

As more effort is required to develop complex software, we expected a negative impact of software complexity on productivity. In addition, complexity affects the extent and difficulty of testing and inspections, thus potentially impacting the success of quality control efforts. Consequently, project complexity seemed likely to affect the quality of the software.

3.2.5. Project size

Software productivity may increase or decrease with project size. Some studies have suggested that the relationship between project size and quality is positive [1], while others have found find that it is negative [6]. Thus size had to be included in our model.

3.2.6. Personnel capability

Software production is labor intensive. Other than the technical factors, experience and capability of the development personnel is very likely to affect the software production process. Krishnan et al.'s analysis suggested that personnel capability had a positive and significant impact on product quality. Therefore, we considered the impact of staff capability on software quality and productivity in our model.

4. Data collection and analysis

4.1. Dataset

We obtained data on 60 business application projects from the database of QAI India to validate our research hypotheses empirically. QAI India is a Consortium for IT Benchmarking; it maintains a repository of quantitative data from Indian software projects to provide a facility for benchmarking them against other projects. Members of the Consortium include leading Indian software organizations; Table 2 provides a summery of industry and project characteristics. QAI keeps a database of basic

Table 2
Project characteristics.

Industry	43%	Manufacturing/Engineering
	19%	Financial/Services
	8%	Utilities
	8%	Government
	22%	Other
Application type	76%	Business
	13%	Telecom
	8%	System
	2%	Scientific
	1%	Other
Platform	51%	Windows
	19%	Unix
	15%	Solaris
	10%	Other
	5%	N/A

information about each member organization, who submit data to the Consortium on the projects that it wants to benchmark.

The project data collection form used by QAI is divided into two sections – core measurements and environment factors. The core measurement section captures the basic information about each project, including project size, amount of development effort, and number of defects found in the testing stage and reported by customers. Such information is provided from the software developer's project database. The environment factors section gathers the project manager's evaluations on the use of development tools, the degree of use of the development methodologies, capability of the personnel, and the technical complexity of the project. The manager rated each item using a Likert-type scale from 0 to 10. To ensure the integrity of the collected data, QAI validates all submitted data by using its own consultants and external experts before placing it into the repository. Our dataset included software projects that were completed by different organizations over a 3-year period. Consequently, our analysis was expected to provide, more generalizable results than analyses using data collected from a single research site.

4.2. Variable definitions

1. We first defined the variable *SIZE* as the project output in thousands of lines of code (KLOC): this is the number of executable lines in the project when it is released. Building upon *SIZE,* we then constructed the variables to measure the three major constructs in our model.
2. *Development productivity* (*DEVPRDTV*) is the ratio of software size and the total effort during the software development process: KLOC divided by development effort in person-months. The development effort included time spent in planning, systems analysis and design, implementation, and testing before product release. The KLOC-based productivity measure allowed for productivity comparison across programming languages, because a developer's production output is not affected by the programming language used.
3. *Development quality* (*DEVQLT*) was the inverse defect rate based on all the defects reported during all the inspection and testing processes before release. The number of defects discovered before release is an indicator for the quality of the code during development. Therefore, we defined it as the project size divided by the total number of defects reported before release.
4. *Product quality* (*PRODQLT)* was the inverse defect rate based on the number of defects reported by the customers in the 12 months period after the product release; this is an indicator of the quality of the software at the time of release. We defined it as the project size divided by the number of defects reported by customers within the 12 months period after release.

We also included five variables to capture other quality and productivity factors:

1. *Design resources* (*DESIGN*) assessed the relative resources spent on planning, as well as on systems analysis and design during the development process, measured as the ratio of the effort (in person-month) exerted on pre-implementation activities over the total development effort (in person-month).
2. *Tool capability* (*TOOLCAP*) assessed the level of tool coverage and the degree of tool integration [3]; it was computed from four items in the QAI data set: tool functionalities in diagramming, database conversion, quality assurance, and the overall integration level of the tools.
3. *Development standard* (*STANDARD*) also was assessed by using four items: the robustness of the adopted development standard, its adaptability, and the development team's experience with and adherence to the standard.
4. *Project complexity* (*COMPLEX*) incorporated three dimensions: component, coordinative, and dynamic complexity [5]. Component complexity referred to the number of elements and requirements of the project; coordinative complexity involved how a project interfaced with other systems, and software dynamic complexity related to the volatility of the project, i.e., the frequency of requirements changes, usually initiated by customers; therefore, we selected four items that captured software complexity in terms of coding and testing requirements, systems interfaces, changes, and customer interfaces.
5. *Personnel capability* (*PERCAP*) was evaluated from both technical and organizational perspectives, using four items that indicated the development team's technical knowledge and skill, group cohesiveness, and overall capabilities.

We performed Confirmatory Factor Analysis (CFA) to determine whether *the selected items were* good indicators of their corresponding constructs (see Table 3). The standardized factor loadings of all the items were greater than 0.4 and significant at the 1% level. The ratio of chi-square to the degrees of freedom fell in the acceptable range of 2–5. The non-normed fit index (NNFI) was greater than 0.80. The Cronbach's Alpha coefficients of the four potential software quality and productivity factors were all greater than 0.70, demonstrating an acceptable level of reliability. Therefore, the overall fit of the measurement model was acceptable. We derived four variables → *TOOLCAP, COMPLEXITY, STANDARD,* and *PERCAP* by taking the averages of the items. In Table 4, we show the descriptive statistics of all the variables in our empirical model.

4.3. Model specification

4.3.1. Linear versus nonlinear specification

Our research model needs a set of three equations for empirical testing. We assumed that *develo*pment quality and development productivity were functions of the six quality factors. Product quality was assumed to be affected by development quality as well as the six quality factors.

Previous literature has suggested a log-linear relationship between the explanatory factors and software quality and software productivity. In keeping with this, we adopted a log-linear specification for our model. We also felt that the relationships between the explanatory and dependent variables was nonlinear. The three equations in our model were therefore:

$$\begin{aligned}\ln DEVQLT = {} & \alpha_0 + \alpha_1 \ln STANDARD + \alpha_2 \ln TOOLCAP \\ & + \alpha_3 \ln DESIGN + \alpha_4 \ln COMPLEX + \alpha_5 \ln SIZE \\ & + \alpha_6 \ln PERCAP + \varepsilon_1 \end{aligned} \quad (1)$$

Table 3
List of survey items used for the main constructs. All the items below are scored on a 0 to 10 point Likert-type scale with 0 being none, 1 being the minimal level, 5 being the average, and 10 being the highest level.

Construct	Question	Standardized factor loading	Composite reliability index
Development standard (*STANDARD*)	What is the robustness of the development standard (0 = no standard)?	0.75	0.76
	What is the level of adherence to the development standard?	0.50	
	What is the level of experience with the development standard?	0.61	
	What is the level of adaptability of the development standard in handling different size systems?	0.86	
Tool capability (*TOOLCAP*)	What is the capability of the diagramming tools?	0.48	0.71
	What is the capability of the quality assurance tool?	0.52	
	What is the capability of the database conversion utilities?	0.53	
	What is the level of integration of your tools?	0.70	
Project complexity (*COMPLEX*)	What is the volume of expected requirement changes?	0.57	0.70
	What is the level of complexity anticipated in interfacing with the customer?	0.48	
	What is the level of complexity in integrating and testing the code?	0.46	
	What is the level of complexity in interfacing with external systems?	0.89	
Personnel capability (*PERCAP*)	What is the overall level of capability of the development team?	0.74	0.81
	What is the level of functional knowledge of the development team?	0.82	
	What is the availability of skilled staff?	0.83	
	What is the level of cohesiveness of the development team?	0.50	

$$\ln DEVPRDTV = \beta_0 + \beta_1 \ln DEVQLT + \beta_2 \ln STANDARD + \beta_3 \ln TOOLCAP + \beta_4 \ln DESIGN + \beta_5 \ln COMPLEX + \beta_6 \ln SIZE + \beta_7 \ln PERCAP + \varepsilon_2 \quad (2)$$

$$\ln PRODQLT = \gamma_0 + \gamma_1 \ln DEVQLT + \gamma_2 \ln STANDARD + \gamma_3 \ln TOOLCAP + \gamma_4 \ln DESIGN + \gamma_5 \ln COMPLEX + \gamma_6 \ln SIZE + \gamma_7 \ln PERCAP + \varepsilon_3 \quad (3)$$

To validate our choice of log-linear specification, we tested our log-linear model against a linear specification. For each equation, the linear specification was rejected at the conventional levels when we evaluated it against its log-linear alternative. Therefore, the nonlinear relationship between various explanatory variables and software quality or productivity was the better fit.

4.3.2. Testing for simultaneous relationships

Eq. (2) suggests that development quality may impact development productivity. One might also argue that development productivity affects development quality, in which case development quality and productivity could be interdependent. Moreover, it is also possible that development productivity could affect product quality. Although no formal theory supports such claims, we empirically tested for the presence of possible simultaneous relationships between development productivity, development quality and product quality by conducting the Hausman specification test. Our result suggested that there was no simultaneity problem. Therefore, the three equations form a recursive system where ordinary least square (OLS) can be applied for appropriate estimation.

Table 4
Descriptive statistics.

Variable name	Mean	Standard deviation
DEVQLT	1.64	5.54
DEVPRDTV	2.20	5.18
PRODQLT	6.69	6.15
STANDARD	7.50	1.29
TOOLCAP	4.85	1.88
DESIGN	18.10	20.30
COMPLEX	5.00	1.71
SIZE	118.00	276.00
PERCAP	6.93	1.23

4.3.3. Model estimation

We estimated the three equations using OLS, conducting several tests to ensure that our analysis did not violate basic econometric assumptions. We performed two diagnosis tests for multicollinearity. First we examined the variance inflation factors of all the explanatory variables in all the equations; they were all less than 10, indicating multicollinearity was not a problem. Then we examined the condition indices and variance-decomposition matrix. For each condition index greater than 30, we found only one variable with variance decomposition value above 0.90. Consequently, we concluded that multicollinearity was not of concern in our analysis. The Shapiro and Wilk test revealed that the residuals from the three equations did not violate the normality assumption. We show the OLS estimation results for the equations in Table 5.

5. Results and discussion

5.1. Impacts of development quality on development productivity and product quality

Overall, our results demonstrate the benefits of increasing development quality. Our results showed that both the estimated coefficients of ln *DEVQLT* were significant at the 1% level, supporting our research hypotheses H1 and H2. Thus an improvement in development quality leads to increases in both development productivity and product quality while all the other variables in the equations remain constant. Therefore, the significant contributions of a high-quality development process to high development productivity and product quality were confirmed.

Table 5
Estimation results for the models of software productivity and quality.

	ln *DEVQLT* Adjusted $R^2 = 0.19$	ln *DEVPRDTV* Adjusted $R^2 = 0.58$	ln *PRODQLT* Adjusted $R^2 = 0.27$
ln *DEVQLT*		0.31 (3.46***)	0.44 (4.46***)
ln *STANDARD*	2.21 (1.46*)	−0.70 (−0.70)	−1.06 (−0.96)
ln *TOOLCAP*	0.67 (1.34*)	0.29 (0.88)	−0.16 (−0.44)
ln *DESIGN*	0.06 (0.44)	0.00 (−0.01)	0.19 (1.97**)
ln *COMPLEX*	0.54 (0.87)	−0.92 (−2.29**)	−0.13 (−0.30)
ln *SIZE*	0.31 (3.16***)	0.43 (6.16***)	0.00 (0.02)
ln *PERCAP*	−0.46 (0.34)	0.55 (0.63)	−0.06 (−0.06)

t-Statistics in parenthesis.
* Indicate statistical significance at 10% levels (one-tailed).
** Indicate statistical significance at 5% levels (one-tailed).
*** Indicate statistical significance at 1% level (one-tailed).

Since the customer only sees the final product, a high product quality level may appear to be all that is desired. Our model showed that product quality can be achieved by a development process that generates low-error code from the start. Such a process does not rely heavily on the testing phase to ensure a high quality product. Therefore, investments in a process that improves the quality of the code production may pay off not only through an improvement in product quality, but also through an increase in productivity.

In addition, in our multiplicative model specification, the value of the estimated coefficient of development quality in the development productivity equation is less than 1 ($|\beta_1| < 1$), indicating the positive effect that improvement of development quality had on development productivity decreased at higher levels of development quality. The relationship between development quality and development productivity is illustrated in Fig. 2, which is based on our estimation results of Eq. (2) with all the other variables in the equation introduced at their sample mean values. In a similar manner, the result $|\gamma_1| < 1$ suggested that the positive effect that improvements of development quality had on product quality decreased at higher levels of development quality. The relationship between development quality and product quality is demonstrated in Fig. 3, which is based on our estimation results of Eq. (3) with all the other variables in the equation introduced at their sample mean values. Thus the marginal benefits of development quality improvement decreased at higher levels of development quality and initial development quality improvement benefitted development productivity and product quality more than improvements to already high-quality processes. A threshold development quality level may exist such that any development quality improvement beyond this level would not be justified in terms of development productivity and product quality improvements.

Previous studies have shown some aspects of the productivity–quality relationship. They demonstrated that an increase in development quality resulted in a decrease in cycle time, development effort and infrastructure cost, while an increase in product quality was related to an increase in life-cycle productivity. Our study combined the different quality constructs (development and product quality) in a set of models, suggesting that an increase in development quality can improve development productivity and product quality. Knowing the quality benefit in terms of development productivity is useful for managers, because this measure allows appropriate comparison of project performance. Our results also demonstrated that development quality is a critical determinant of product quality, thus it is essential to enhance development quality in order to increase product quality.

5.2. Drivers of development quality

As we confirmed the contributions of development quality to development productivity and product quality, we also examined the factors that can enhance development quality. Our empirical results for development quality Eq. (1) showed that the estimated coefficients of development standard, tool capability and project size were significant at conventional levels, while those of design resources, personnel capability, and project complexity were not. This suggested that an increase in process standardization, tool capability, or project size could significantly increase the level of development quality.

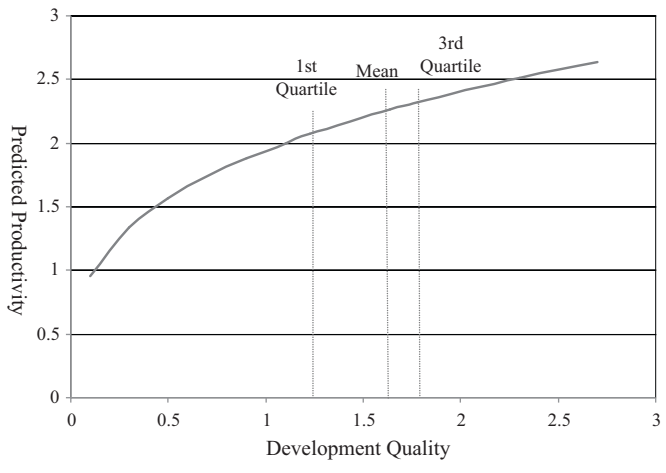

Fig. 2. The relationship between Development Quality and Development Productivity.

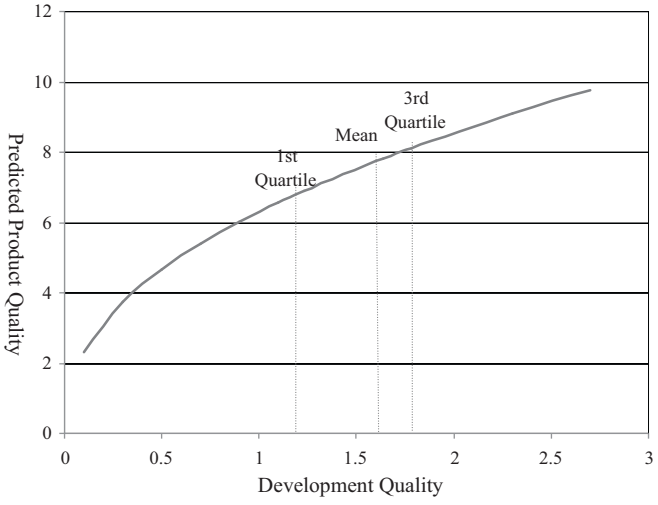

Fig. 3. The relationship between Development Quality and Product Quality.

The significant and positive relationship between process standardization level and development quality level confirmed the benefits of process standardization promoted by CMM and similar initiatives: higher process maturity levels *do* lead to higher development quality. The significant and positive impact of tool capability on development quality demonstrated the contribution of tool adoption in improving the software development process: they may allow a semi-automated software development process, thereby introducing fewer errors in the code. The significant and positive relationship between project size and development quality indicated that, for our dataset, larger projects tended to have less defects identified during the development process; it is possible that developers are more careful when they work on large projects.

5.3. Additional factors

After considering the contribution of development quality, only a few factors affected product quality and development productivity: ln *DESIGN* was the only significant independent variable in Eq. (3). An increase in the proportion of resources dedicated to design activities was also found to lead to an increase in product quality. Utilizing more resources for project planning, analysis, and design could also improve product quality. Therefore, a well-designed project can better meet the customer's expectation of product quality.

In contrast, development standard, complexity, tool capability, project size, and personnel capability do not have a significant impact on product quality after development quality is considered. This does not, however, denigrate the value of improving development standard and tool capability for enhancing product quality: it implies that these factors may have indirect impacts on product quality through changes in development quality.

In addition to development quality, project size and complexity are the only two factors affecting development productivity. The significant and positive effect of project size on development productivity suggests the existence of economies of scale within our sample. The insignificant impact of development standard and tool capacity on development productivity are consistent with those reported by Krishnan et al. Since higher standardization level and higher tool capability requires additional effort to learn and follow disciplined practices and tool application, we believed that the positive influence of standardization and tool on productivity may be offset by the additional development cost. It is also possible that the development standard and tool capability have indirect impacts on development productivity through development quality.

Personnel capability did not prove to be significant in the results of any of the three models. However, when high level of development standard is in place (the average value in our sample was 7.5), it may dominate other factors. This is analogous to results discussed by Agrawal and Chari [1]: they showed that in CMM Level 5 projects, factors other than project size were insignificant in determining development effort and product quality.

6. Conclusion

Our research investigated the role of quality in software development. We developed a research framework depicting the relationships between development quality, development productivity, and product quality. The empirical analysis introduced and validated a productivity model, as well as two separate models of development quality and product quality, using data collected on software projects performed in India.

The results demonstrated that development quality is the key to successful and cost-effective software production. The benefits of development quality improvements are twofold: better product quality and higher productivity. Apparently, only a few factors have impact on product quality and development productivity. Therefore, software developers should aim for high quality (few defects) during code production. The models also showed that the major factors that impact development quality include the use of a development standard, good tool capability, and the size of the project. Further, we demonstrated the link between a software developer's internal view of software quality (development quality) and the development productivity and software quality realized by the client (product quality).

These findings are highly relevant to software managers, as they demonstrated the positive effect of investment on the cost and quality of the development process. Our study was unique in that it used data on projects from a large number of organizations. The use of such comprehensive data for the empirical validation of the productivity–quality model enhanced the generalizability of our results. Data on benchmarking projects were used to validate the proposed model. Because organizations typically enter such projects voluntarily, they may have selected their best projects for inclusion. While this should not affect the validity of the quality and productivity relationships, this could bias our results and should be considered as a limitation.

References

[1] M. Agrawal, K. Chari, Software effort, quality, and cycle time: a study of CMM level 5 projects, IEEE Transactions on Software Engineering 33 (3), 2007, pp. 145–157.
[2] N. Ashrafi, The impact of software process improvement on quality: in theory and practice, Information and Management 40 (7), 2003, pp. 677–690.
[3] J. Baik, B. Boehm, B.M. Steece, Disaggregating and Calibrating the CASE Tool Variable in COCOMO II, IEEE Transaction on Software Engineering 28 (11), 2002, pp. 1009–1022.
[4] J.W. Bailey, V.R. Basili, A meta-model for software development resource expenditures, in: Proceedings of the Fifth International Conference on Software Engineering, San Diego, CA, 1981.
[5] R.D. Banker, G.B. Davis, S.A. Slaughter, Software development practices, software complexity, and software maintenance performance: a field study, Management Science 44 (4), 1998, pp. 433–450.
[6] R.D. Banker, S.A. Slaughter, The moderating effects of structure on volatility and complexity in software enhancement, Information Systems Research 11 (3), 2000, pp. 219–240.
[7] B.W. Boehm, Software Engineering Economics, Englewood Cliffs, NJ, Prentice Hall, 1981.
[8] E. Capra, C. Francalanci, F. Merlo, An empirical study on the relationship among software design quality, development effort, and governance in open source projects, IEEE Transactions on Software Engineering 34 (6), 2008, pp. 765–782.
[9] L.Y. Fok, W.M. Fok, S.J. Hartman, Exploring the relationship between total quality management and information system development, Information and Management 38 (6), 2001, pp. 355–371.
[10] E.D. Harter, M.S. Krishnan, S.A. Slaughter, Effects of process maturity on quality, cycle time, and effort in software product development, Management Science 46 (4), 2000, pp. 451–466.
[11] E.D. Harter, S.A. Slaughter, Quality improvement and infrastructure activity costs in software development: a longitudinal analysis, Management Science 49 (6), 2003, pp. 784–800.
[12] S.-J. Huang, W.-M. Han, Selection priority of process a areas based on CMMI continuous representation, Information and Management 43 (3), 2006, pp. 297–307.
[13] G. Issac, C. Rajendran, R.N. Anantharaman, Significance of quality certification: the case of the software industry in India, Quality Management Journal 11 (1), 2004, pp. 8–32.
[14] J.J. Jiang, G. Klein, H.G. Hwang, J. Huang, S.-Y. Hung, An exploration of the relationship between software development process maturity and project performance, Information and Management 41 (3), 2004, pp. 279–288.
[15] M.S. Krishnan, C.H. Kriebel, S. Kekre, T. Mukhopadhyay, An empirical analysis of productivity and quality in software products, Management Science 46 (6), 2000, pp. 745–759.
[16] A. MacCormack, C.F. Kemerer, M. Cusumano, B. Crandall, Trade-offs between productivity and quality in selecting software development practices, IEEE Software 2003, pp. 78–85.
[17] K.D. Maxwell, L. Van Wassenhove, S. Dutta, Software development productivity of european space, military, and industrial applications, IEEE Transactions on Software Engineering 22 (10), 1996, pp. 706–718.
[18] J.S. Osmundson, J.B. Michael, M.J. Machniak, M.A. Grossman, Quality management metrics for software development, Information and Management 40 (8), 2003, pp. 799–812.

[19] T. Ravichandran, A. Rai, Quality management in systems development: an organizational perspective, MIS Quarterly 24 (3), 2000, pp. 281–415.
[20] I. Sila, M. Ebrahimpour, C. Birkholz, Quality in supply chains: an empirical analysis, Supply Chain Management 11 (6), 2006, pp. 491–502.
[21] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben, Y. Liu, Productivity factors and programming environments, in: Proceedings of the Seventh International Conference on Software Engineering, Orlando, FL, 1984.
[22] C.E. Walston, C.P. Felix, A method of programming measurement and estimation, IBM Systems Journal 16 (1), 1977, pp. 54–73.

Marcus A. Rothenberger is an Associate Professor and the Chair of the Department of Management Information Systems at the University of Nevada Las Vegas (UNLV). He holds a Ph.D. in Information Systems from Arizona State University. He uses qualitative, quantitative, and design science paradigms to research software process improvement issues, software reusability, performance measurement, service-oriented architectures, and the adoption of enterprise resource planning systems. His work has appeared in major academic journals, such as the *Journal of Management Information Systems*, the *Decision Sciences Journal*, *IEEE Transactions on Software Engineering*, IEEE *Transactions on Engineering Management*, *Communications of the ACM*, and *Information & Management*. He is a Co-Editor-in-Chief of *the Journal of Information Technology Theory and Application (JITTA)* and an Associate Editor of the *AIS Transactions on Enterprise Systems*. He is a member of the Association for Information Systems.

Yi-Ching Kao is an Assistant Professor of Accounting at Menlo College. She received her Ph.D. in Management Science from the University of Texas at Dallas. Her current research interest is performance evaluation in various services industries including public accounting, software production, electronic commerce and non-profit organizations. She has published in the *European Journal of Information Systems*, *Decision Support Systems*, and the *Journal of Information Systems*.

Luk Van Wassenhove is Professor at INSEAD and holds the Henry Ford chair in Manufacturing. Before joining INSEAD, he was on the faculty at Erasmus University and at the Katholieke Universiteit Leuven His research and teaching are concerned with operational excellence, supply chain management, quality, continual improvement and learning. His recent research focus is on closed-loop supply chains (product take-back and end-of-life issues) and on disaster management (humanitarian logistics). He is departmental editor for *Production and Operations Management*, and associate editor for *Technology and Operations Review* and *International Journal of Production Economics*. He publishes extensively in *Management Science*, *Operations Research*, *The International Journal of Production Research*, *The European Journal of Operations Research* and in many other academic as well as management journals (*Harvard Business Review*, *California Management*