Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects

Manish Agrawal and Kaushal Chari

Abstract—The Capability Maturity Model (CMM) has become a popular methodology for improving software development processes with the goal of developing high-quality software within budget and planned cycle time. Prior research literature, while not exclusively focusing on CMM level 5 projects, has identified a host of factors as determinants of software development effort, quality, and cycle time. In this study, we focus exclusively on CMM level 5 projects from multiple organizations to study the impacts of highly mature processes on effort, quality, and cycle time. Using a linear regression model based on data collected from 37 CMM level 5 projects of four organizations, we find that high levels of process maturity, as indicated by CMM level 5 rating, reduce the effects of most factors that were previously believed to impact software development effort, quality, and cycle time. The only factor found to be significant in determining effort, cycle time, and quality was software size. On the average, the developed models predicted effort and cycle time around 12 percent and defects to about 49 percent of the actuals, across organizations. Overall, the results in this paper indicate that some of the biggest rewards from high levels of process maturity come from the reduction in variance of software development outcomes that were caused by factors other than software size.

Index Terms—Cost estimation, time estimation, software quality, productivity.

1 Introduction

Developing software to meet functional needs with acceptable levels of quality, within budget, and on schedule, is a goal pursued by every software development organization. Many organizations are adopting the best practices in software development, such as those based on Capability Maturity Model (CMM) [1], ISO 9001 [2], or Six Sigma [3]. CMM has been one of the most popular efforts in enhancing software quality and reducing development costs [4], [5], [6].

Although development effort, software quality, and cycle time have been studied in prior research on software estimation [7], [8], [9], [10], most of the published results are based on data sets that are now considered dated (see Table 1). Due to various technological innovations such as the use of object-oriented languages, middleware, and newer tools and due to increased adoption of best practices in software development, that is, those based on CMM, ISO 9001, or Six Sigma, there is a need to reexamine relationships between software project development outcomes and various factors identified from prior literature.

Conventional wisdom suggests that there are conflicting influences on software development effort, quality, and cycle time: Cycle time may be compressed at the cost of quality, experienced professionals may improve quality but at increased costs, quality may be achieved at the cost of increased testing effort, larger team size may reduce development time while raising total costs, process maturity may improve quality but at high cost, and so forth. However, research suggests that one of the most important consequences of improved processes is superior conformance quality [11]. The reduction in variability is likely to be most pronounced in development organizations at CMM level 5, which is the highest level of process maturity as per the Software Engineering Institute (SEI) located at Carnegie Mellon University. To our knowledge, with the exception of a recent study [12], previous studies on software development outcomes have not exclusively focused on CMM level 5 projects. The study in [12] presents models on productivity and conformance quality based on data from a single organization. Its results, therefore, may not be generalizable outside the environment where they were calibrated.

Valuable insights can be gained from a study that focuses exclusively on CMM level 5 software development projects. For example, it would be possible to determine the factors that really matter in determining project development outcomes, as well as the benefits that are accrued, when software development processes are at the highest levels of maturity. Furthermore, benchmarks based on CMM level 5 projects could be useful goals that many non-CMM level 5 software development organizations could strive to achieve for their own projects.

In this paper, we make two major contributions. First, we identify key project factors such as software size that determine software project development outcomes for CMM level 5 projects. Second, we provide benchmarks for effort, quality, and cycle time based on CMM level 5 project data. Our results suggest that estimation models based on CMM level 5 data are portable across multiple CMM level 5 organizations. This paper is a refinement of a conference paper [13] that had results not exclusively based on CMM level 5 data.

• *The authors are with the Department of Information Systems and Decision Sciences, College of Business Administration, University of South Florida, 4202 East Fowler Avenue, CIS 1040, Tampa, FL 33620-7800. E-mail: {magrawal, kchari}@coba.usf.edu.*

Manuscript received 15 May 2005; revised 18 May 2006; accepted 16 Nov. 2006; published online 26 Jan. 2007.
Recommended for acceptance by P. Jalote.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0332-1205.

TABLE 1
Prior Literature

Study	Data Set Year (If reported)	Select Dependent Variables	Select Independent Variables	N	Outcome
Gaffney (1984) [18]		Defects	Size	25	Program size provides good estimates of the number of defects.
Kemerer (1987) [15]	1981-1987	Effort	Size	15	Function point models provide superior effort estimates compared to SLOC models.
Banker and Kemerer (1989) [19]	1970-1980	Effort	SLOC	15-63	Small projects show economies of scale, large projects show diseconomies of scale.
Compton and Withrow (1990) [20]		Defects	Size, Complexity Metric	263 packages of a system	Goldilocks principle – intermediate-sized packages have the least defects.
Mukhopadhyay, Vicinanza and Prietula (1992) [8]	1981-1987	Effort	Function counts, Project attributes	15	Case-based reasoning provides superior effort estimates than COCOMO.
Matson, Barrett and Mellichamp (1994) [21]	-1994	Effort	Size (in FP)	104	Effort has a log-linear relationship with size in FP.
Diaz and Sligo (1997) [6]	1992 and before	Defect Density, Cycle-Time, Productivity	CMM (4-5)	34	Each CMM level improves quality by a factor of 2, generally reduces cycle-time and, to a lesser extent, also improves productivity.
Shepperd and Schofield (1997) [22]		Effort	Features	275	Analogies provide superior effort estimates.
Krishnan and Kellner (1999) [23]	1993-1995	Quality	Consistent adoption of CMM practices, Size	45	Consistent adoption of CMM processes reduces defects.
Maxwell, Wassenhove and Dutta (1999) [24]	1988-	Effort	Size, Language, Reliability	108	Firm-specific models provide superior effort estimates than general models.
Boehm et. Al. (2000) [7]	-1998	Effort	Size, maturity, experience	161	Size and product complexity are the most significant drivers of effort
Clark (2000) [25]		Effort	Equivalent process maturity (Self assessed)	161	One level change in maturity reduced effort by 3%-15%.
Harter, Krishnan and Slaughter (2000) [4]	1984-1996	Effort, Quality, Cycle-Time	CMM (1-3), Size	30	CMM improvements reduce cycle-time and effort.
Krishnan, Kriebel, Kekre and Mukhopadhyay (2000) [26]	1993-1998	Quality, Life-Cycle-Productivity	Personnel capability, Size, Process factors	43	Personnel capability and process factors improve quality, and quality improves life-cycle productivity.
Baik (2002) [16]	1970's and 1980's	Effort	Tool use	15	Disaggregation of TOOL variable in COCOMO II improves prediction accuracy from 67% to 87%.
Harter and Slaughter (2003) [9]	1985-1994	Quality	CMM (1-3)	9 infrastructure activity centers in one firm	CMM improvements reduce defect density.
Nan, Harter and Thomas (2003) [27]	1984-1994	Effort, quality, cycle-time	Schedule pressure		Schedule pressure can reduce effort and cycle-time without sacrificing quality.
Ramasubbu, Mithas, Krishnan, and Kemerer (2004) [12]		Productivity, Quality	Work Dispersion, Defect Prevention	42	Work Dispersion has a negative effect on quality and productivity in a CMM level 5 organization. Appraisal-based quality management improves productivity while failure-based quality management improves conformance quality.
Pendharkar, Subramanian and Rodger (2005) [28]	-1996	Effort	Development methodology, tool usage	33	Bayesian models provide competitive effort estimates.

The results in this paper are based on data collected from 37 projects of four CMM level 5 organizations. The application domains of sample projects primarily pertain to the general category of management information system (MIS)/business applications, with 34 out of 37 projects reported in this category. Start dates for the projects in our sample ranged from January 1998 to September 2004, and end dates ranged from May 2001 to October 2004. Software development data is very difficult to obtain in the best of circumstances, and the sample size of the 37 projects used in the study is comparable to other published studies. For example, the sample sizes are 30 in [4], 24 in [14], 15 in [15],

and 15 in [16]. The four organizations chosen in our study represent a convenience sample from a population of 141 CMM level 5 organizations worldwide [17]. Table 1 summarizes the data sets and conclusions from a selected set of prior studies.

The rest of the paper is organized as follows: A review of relevant literature is presented in Section 2. The research model, design, and methodology used are presented in Section 3. Section 4 contains data analysis and results, whereas conclusions are presented in Section 5. Appendix A contains the significance levels of all independent variables.

2 Literature Review

We first summarize the impacts of the factors from prior research, which have been used to estimate development effort, cycle time, and quality. A summary of relevant literature is in Table 1.

2.1 Software Development Effort

Software development effort typically includes human effort expended for high-level design, detailed design, coding, unit testing, integration testing, and customer acceptance testing. Effort is often regarded as a surrogate for software development cost since personnel cost is the dominant cost in software development.

Many models such as COCOMO [7], PRICE-S [29], ESTIMACS [30], SEER-SEM [31] have been developed to estimate software development costs. Effort-estimation models such as COCOMO primarily use the number of source lines of code (SLOC) as the basis for effort estimation [32]. Thus, effort in man-months is expressed as a function of kilo source lines of code (KSLOC). The COCOMO II model, which is the current version of COCOMO, uses 17 effort multipliers and five scale factors to estimate development effort based on project size. Some of these effort multipliers such as application experience (APEX) and language and tool experience (LTEX) have been found to be insignificant [7]. An alternative metric for SLOC is function points (FPs) [33], where the FP is the product of the number of function counts and the processing complexity adjustment [21].

An excellent summary of early models to estimate software development effort has been provided by Mohanty [34]. For a software system with 36,000 lines of machine language executable instructions and well-defined specifications for all independent variables, the various models described in [34] have predicted costs ranging from $300,000 to $2,500,000 and development times ranging from 13 to 25 months. Kemerer [15] compared software estimation models such as COCOMO [7], SLIM [35], Function Points [14], and ESTIMACS [36] using data from 15 projects with an average size of a little under 200 KSLOC and found that various estimation models resulted in average effort estimation error rates ranging from 85 to 772 percent. This wide range has been attributed to the differences in productivity between the test environment and the environments in which the models were calibrated, suggesting wide variations in software development outcomes across organizations. Also, differences in application domain influenced the accuracy of these estimates. For example, the projects in the data set used in [16] were primarily business applications with 12 out of 15 projects implemented in COBOL. In contrast, the COCOMO database consisting of 63 projects had only seven business applications [32].

A study by Maxwell et al. [24] found that a relatively small set of factors explained the required effort to complete a software project (size in SLOC), and productivity factors such as application category, language, required reliability, and programming practices. This study also found that organization-specific models predicted required effort more accurately than general models. It was therefore important to identify organization-level factors that affected software development costs. Banker and Slaughter [37] found that data complexity, defined as the number of data elements per unit of application functionality, significantly increased the enhancement cost of software.

Specifically focusing on the impact of capability maturity, improvements in process maturity were found to be associated with reductions in effort [4], [25]. According to an SEI report [38], by adopting Capability Maturity Model Integration (CMMI)-based process improvements, Boeing Australia had a 60 percent reduction in work, whereas Lockheed Martin achieved a 30 percent increase in software productivity. Although an increase in the CMM level had a direct effect of increasing development effort, the associated improvements in quality reduced development effort, presumably through reduced rework and improved understanding of software requirements. Another study [26] found that process improvements were not significantly related to development costs. Perhaps reflecting on the lack of theory on software estimation, a number of studies found success at effort estimation by simply using analogies to compare the features of a new project with earlier projects [8], [22], [39], [40].

2.2 Software Quality

The definition of software quality has evolved over time [41]. Initially, it was defined as conformance to a standard or a specification. Later, the definition was changed to adapt to highly dynamic business environments. In 1991, the International Organization for Standardization adopted ISO 9126 as the standard for evaluating software quality. This standard defines quality as "the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs" [42]. ISO 9126 compliments ISO 9001, which deals with the quality assurance of the process used for developing products. A commonly used definition of software quality is the density of postrelease defects in a software program, which is measured as the number of defects per thousand lines of code [6], [43].

Gaffney [18] reported that the best estimator for the number of errors in a software module was the number of lines of code. Krishnan and Kellner [23] also confirmed this finding. Harter and Slaughter [44] found that product complexity significantly lowered software quality, which is somewhat contrary to [18], which did not find software complexity affecting error rates significantly.

Banker and Slaughter [37] found software volatility, defined as the frequency of enhancements per unit of functionality in a given time frame, to be a significant

predictor of software errors. Data complexity, defined as the number of data elements per unit of application functionality, also increased the number of defects. They also found that structured programming techniques moderated the effects of volatility and data complexity on software errors. Using a game-theoretic model, Austin [45] suggested that under schedule pressures, developers were likely to compromise on quality. Krishnan et al. [26] found personnel quality, which is measured using peer and supervisor assessments, to be a significant estimator of software quality. They also found that front-end investments, which improved customer-requirements analysis, enhanced quality.

A number of approaches have been proposed to improve software quality [46]. These include total quality management (TQM), Six Sigma [3], and CMM [1]. The basic idea behind all these approaches is to identify ways to improve quality in a given situation. The relationship between process improvements and quality has also been investigated [47]. The most significant development in this area has been the development of CMM [2], [48]. For example, as a software unit at Motorola improved from CMM level 2 to level 5, the average defect density reduced from 890 defects per million assembly-equivalent lines of code to about 126 defects per million assembly-equivalent lines [6]. In an empirical study using 33 software products developed over 12 years by an IT company, Harter et al. [4], [44] found that a 1 percent improvement in process maturity was associated with a 1.589 percent increase in product quality. In another study, Krishnan and Kellner [23] found process maturity and personnel capability to be significant predictors (both at the 10 percent level) of the number of defects.

2.3 Cycle Time

Cycle time is an important outcome variable because software projects are often carried out under strict delivery schedules. The cycle time for software development, which depends on two factors (planned development time and discrepancies between planned and actual development times) can be tracked by an Earned Value Management System (EVMS). When planned schedules are longer than the minimum cost-effective schedule, they do not raise development costs. This is because, under ideal circumstances, projects can be completed using fewer developers than the optimal staffing strength. However, when schedules get tighter, cycle time can be adversely affected by dysfunctional team dynamics as per Brooks' Law [49].

In an empirical study that explored discrepancies between planned and actual durations, it was found that the most significant reasons for these discrepancies were capacity-related issues, such as personnel involved in unplanned activity, maintenance of earlier projects, and so forth, and product-complexity related issues such as underestimating the application complexity [50].

The number of lines of code has been reported to be a significant predictor of construction time [4]. The level of process maturity also has some bearing on the development time. Based on the measurements at Motorola, Diaz and Sligo [6] reported that at CMM level 5, the cycle time was about eight times faster than at CMM level 1. Harter et al. [4], [51] also reported similar time savings from process improvements. This is attributed to the reduction in rework due to improved processes thereby leading to reduced cycle time.

2.4 The Impact of Process Improvement

Process management is one of the most significant management innovations in recent times. Process management focuses on increasing process control in order to reduce variances in outcomes. The final step in a process improvement program involves the adoption of standardized best practices throughout the organization and the continuous refinement of these best practices [52]. Accordingly, a CMM level 5 process is characterized as an "optimizing process," that is, a quantitatively managed process (CMM level 4) that is changed and adapted to meet relevant business objectives. One of the major goals of this study is to examine whether process improvements from level 5 maturity reduce variance in software development outcomes.

3 Research Model, Design, and Methodology

It can be seen from the previous section that prior research on software process improvement has focused on finding evidence of reduced effort, improved quality, and faster cycle times from software process improvements. In this paper, we examine the impacts of some of the most important factors identified from prior research on software development effort, quality, and cycle time, while focusing specifically on CMM level 5 projects. The various variables used in the study are described next.

3.1 Development Effort

We define software development effort as the total effort beginning with the end of the requirements specification stage until the end of customer acceptance testing. It includes effort during high-level design, detailed design, coding, unit testing, integration testing, and customer acceptance testing. EFFORT represents software development effort and is measured in person-days.

3.2 Product Quality

The metric used for product quality (QUAL) is defects, which were measured as the total number of defects that escaped to the customer and were detected during the first three months of production use of the software. A period of three months is used, as it is typically the warranty period of newly developed software, and the defect data for the first three months is generally tracked by software development organizations. A low value of QUAL indicates higher quality software.

Although defect density or its reciprocal, which normalize defects for size, have been used as a measure of quality in prior literature [6], [26], we use total number of defects instead due to high correlation between defect density and size. When we do log transforms of variables in our analysis,

$$\ln(\text{defect-density}) = \ln(\text{defects}/\text{size}) = \ln(\text{defects}) - \ln(\text{size}),$$

we find that $\ln(\text{defect-density})$ has a correlation of -0.86 with $\ln(\text{size})$. Since this correlation is very high and highly

significant, using quality and size together as independent variables in the same model causes multicollinearity problems. Our approach of using total number of defects as a measure of quality is in line with prior research [23].

3.3 Cycle Time

Cycle time, represented by CTIME, is measured as the number of calendar days that elapse from the date the baseline requirements are delivered until the date the software passes customer-acceptance testing.

3.4 Product Size

Product size (SIZE) can be measured using lines of codes or using FPs. Lines of code count the actual number of lines in the application, and FPs count the number of functional units. In this paper, we use actual lines of codes developed, excluding comments and blank lines, measured in KSLOC to represent product size. Although the use of KSLOC is in line with prior research [23], a limitation of this measure is that it is usually not consistent across programming languages. When size data is only available in FPs, we use backfiring that uses a conversion table in [7, p. 20] to convert FPs to lines of code. FPs are computed based on International Function Point Users Group (IFPUG) specifications. In Section 4, we report the results with and without observations on FPs.

3.5 Product Complexity

Product complexity (COMPLX) is measured using two items on a seven-point Likert scale, ranging from low to high data complexity and decision complexity. Data complexity is defined as the anticipated level of difficulty in development because of complicated data structures and database relationships. Decision complexity is the difficulty in managing decision paths and structures within the product. Overall product complexity is computed as the mean of data complexity and decision complexity.

3.6 Schedule Pressure

Schedule pressure (SP) is defined as the relative compression of the development schedule mandated by management compared to the initial estimate provided by the development team based on project parameters [27]:

$$\mathrm{SP} = (Team\ estimated\ cycle\text{-}time - Management\ mandated\ cycle\text{-}time)/Team\ estimated\ cycle\text{-}time.$$

When schedule pressure is zero, the log of schedule pressure, which is used in log models, is undefined. In such cases, the numerator in the above expression is set to a small value (0.1) so that the log of the schedule pressure can be calculated for use in regression equations for log models.

3.7 Team Size

Software development teams grow and contract at different phases of the project, and the staffing strengths typically follow a Rayleigh curve as a function of time [53]. Therefore, the size of a team (TEAM) at its peak is considered a good proxy for the relative size of the team compared to other projects. Also, the peak team size is easier to measure than the average team size over the life of the team. Therefore, TEAM is measured as the peak team size.

3.8 Personnel Capability

Various measures for representing personnel capability have been used in prior research [7], [54]. These include education levels, industry experience, and subjective ratings of project supervisors and other team members. Based on an extensive review and survey, Nelson [55] identified two dimensions of personnel capability for information systems: technical skills and organization skills. In this paper, we measure the technical skills dimension using five subjective items derived from items reported in [55] using a Likert scale. Supervisors (that is, managers or team leaders) provide a mean assessment of team members on programming languages, analysis and design, computer-aided software engineering (CASE) tools, databases, and data communications technologies. For example, programming language skills is measured using the following question, "*What was the average skill level of your team members in using computer programming languages relevant for the application?*" We use three items to capture team skills, which overlap with organizational skills in [55]. The items measuring team skills capture a supervisor's subjective assessment of the quality of citizenship, cooperation among team members, and overall performance of the team.

The technical skill of each project team is computed as the mean of the five items used to measure the technical capabilities of the team members. Team skill is calculated as the mean of the three items used to measure individual team skills. Finally, we have also included an overall item to obtain the supervisor's average rating of the team members. Personnel capability (PERCAP) is calculated as the mean of technical skills, team skills, and supervisors' overall average rating.

3.9 Requirements Quality

Requirements quality is represented using two variables: requirements volatility and requirement specifications quality. Requirements volatility (REQVOL) is measured as the number of changes made to the baseline specifications during the development phase. This is obtained as a numerical measure from project data. Requirements specifications quality (REQUAL) is calculated as the project supervisor's mean rating for the eight attributes of good requirements (that is, correctness, unambiguity, completeness, consistency, ranking, verifiability, modifiability, and traceability) specified in IEEE Standard 830 [56].

3.10 Project Supervisor Experience

To account for the management skills of project supervisors, we use their experience in the software industry (INDEXP), as well as their experience in managerial roles (MGROL) within the industry. Both measures are used to examine the impact of managerial quality on software outcomes. Unlike the assessment of team members' capabilities, objective measures of supervisor experience are preferred over self-assessments to minimize self-response bias. It is expected that supervisors with greater industry or managerial experience would provide better managerial inputs leading to superior project outcomes.

3.11 Modularity

To measure the impact of increasing complexities arising from the dependencies between software modules, we

TABLE 2
Independent Variables in this Research

Variable (components if any)	Description	Measurement Type
EFFORT	Person-days expended between baseline requirements specifications and customer acceptance	Objective
QUAL	Defects observed during first quarter of use	Objective
CTIME	Calendar days between baseline requirements specification and customer acceptance	Objective
SIZE	Lines of code in the software project (in KSLOC)	Objective
COMPLX	Mean of data and decision complexity in the software project	Subjective
SP	Relative compression of the development time imposed by management compared to initial estimate by development team	Objective
TEAM	Size of the team at its peak	Objective
PERCAP	Mean rating by supervisor of team members' technical skills, team skills and overall rating	Subjective
REQVOL	Number of changes to requirements during development	Objective
REQUAL	Supervisor's assessment of the quality of requirements	Subjective
INDEXP	Supervisor's years of experience in the software industry	Objective
MGROL	Supervisor's years of experience in a managerial role	Objective
MOD	Modular complexity	Objective

measure modular complexity (MOD) as $n(n-1)/2$, where n is the number of modules in the software [57].

These variables are summarized in Table 2.

3.12 Empirical Models

We use the following empirical models for effort, quality, and cycle time. Prior research suggests that effort and cycle time are affected by quality because improved quality reduces rework and reduces effort and cycle time [19]:

$$\mathrm{QUAL} = \mathrm{f}(\mathrm{SIZE}, \mathrm{COMPLX}, \mathrm{SP}, \mathrm{TEAM}, \mathrm{PERCAP}, \mathrm{REQVOL}, \mathrm{REQUAL}, \mathrm{INDEXP}, \mathrm{MGROL}, \mathrm{MOD})$$

$$\mathrm{EFFORT} = \mathrm{f}(\mathrm{SIZE}, \mathrm{QUAL}, \mathrm{COMPLX}, \mathrm{SP}, \mathrm{TEAM}, \mathrm{PERCAP}, \mathrm{REQVOL}, \mathrm{REQUAL}, \mathrm{INDEXP}, \mathrm{MGROL}, \mathrm{MOD})$$

$$\mathrm{CTIME} = \mathrm{f}(\mathrm{SIZE}, \mathrm{QUAL}, \mathrm{COMPLX}, \mathrm{SP}, \mathrm{TEAM}, \mathrm{PERCAP}, \mathrm{REQVOL}, \mathrm{REQUAL}, \mathrm{INDEXP}, \mathrm{MGROL}, \mathrm{MOD}).$$

Prior research suggests that the effects proposed above are not linear and that software development exhibit economies of scale [19]. Therefore, the above relationships are commonly modeled using a log-log transformation as follows. For brevity, SIZE and QUAL are explicitly shown below, whereas the other independent variables are represented by the general representation IV:

$$\ln(\mathrm{QUAL}) = \beta_{10} + \beta_{11}\ln(\mathrm{SIZE}) + \sum_{\mathrm{i}} \beta_{1\mathrm{i}} \ln(\mathrm{IV})_{\mathrm{i}},$$

$$\ln(\mathrm{EFFORT}) = \beta_{20} + \beta_{21}\ln(\mathrm{SIZE}) + \beta_{22}\ln(\mathrm{QUAL}) + \sum_{\mathrm{i}} \beta_{2\mathrm{i}} \ln(\mathrm{IV})_{\mathrm{i}},$$

$$\ln(\mathrm{CTIME}) = \beta_{30} + \beta_{31}\ln(\mathrm{SIZE}) + \beta_{32}\ln(\mathrm{QUAL}) + \sum_{\mathrm{i}} \beta_{3\mathrm{i}} \ln(\mathrm{IV})_{\mathrm{i}}.$$

3.13 Data Collection

We collected project data from four large CMM level 5 organizations in the software development outsourcing industry. These organizations had formal organizational structures dealing with software quality and process improvement issues. Data were obtained under the terms of nondisclosure agreements that placed some restrictions on the information that could be published. Our primary contacts were located in the software engineering process group (SEPG) or equivalent departments of these organizations and were very knowledgeable about the best practices in software development. Our contacts interacted with project supervisors and had access to various tools in the organizational tool repository for managing project related data. Most of these tools were proprietary and developed in-house. Participating organizations reported the use of tools for project management and tracking, defect analysis and reporting, sizing projects, and effort reporting. Two of the organizations also reported using Microsoft Project in project planning. One of the participating organizations reported using employee swipe-in and swipe-out data to cross-check the effort data reported by employees. In addition to using tools for managing projects, participating organizations also had an extensive system of reviews for ensuring the integrity of various project-related data.

With regards to KSLOC size data, all participating organizations reported the count of actual lines of code excluding blank lines and comments. For FPs data (there were a total of eight projects where size was only available in FPs), IFPUG specifications were followed for counting FPs. Organizations were asked to select only those projects where it was easier to report size data attributed to the effort reported. The actual hours spent by project team members were used in effort computations. Out of 37 projects, 34 projects were reported to be in the general category of MIS/business applications. The most common technologies used were Java and Oracle databases. A partial list of other technologies used includes C/C++, .NET, J2EE, PL/SQL, application servers such as WebLogic and WebSphere, and middeware technologies such as MQSeries. Average industry experience of project supervisors was about seven years, of which, about three years was in a managerial role. Mean project size was approximately 360 KSLOC.

To begin with, we collected data on 41 projects. We scrutinized the data thoroughly, and, when there were any discrepancies observed, we contacted the concerned organization for further clarifications. If the clarifications did not resolve the discrepancies satisfactorily, we dropped the project from our sample. Out of an initial sample of 41 projects, we dropped four projects from the sample due to reasons such as incomplete data or inconsistent data.

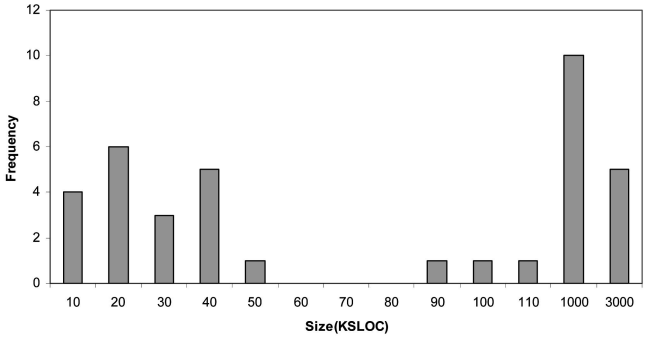

Fig. 1. Distribution of projects by size.

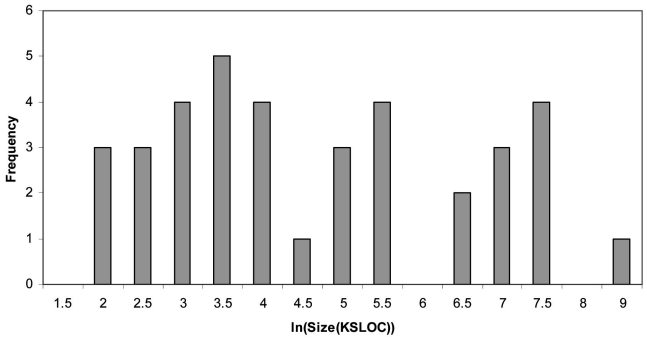

Fig. 2. Distribution of projects by ln(size).

Thus, we ended with a sample of 37 projects for further analysis. Due to the use of various tools and the enforcement of rigorous review procedures in participating organizations, as well as our own efforts in cross-checking data for inconsistencies, we believe that the data is of high quality with minimal biases.

4 Data Analysis and Results

The distributions of raw project sizes and log transformed project sizes are shown in Fig. 1 and Fig. 2, respectively.

The figures show that log transformation improves the distribution of project sizes. To investigate factors that determine project outcomes, we used linear regression with EFFORT, QUAL, and CTIME as dependent variables. All the other variables in Table 2 were used as independent variables. After examining the distribution of project sizes, we decided to transform the variables by taking their log transforms (natural logs) since the distribution by ln(size) was closer to normal. Table 3 provides descriptive statistics for log-transformed independent variables.

TABLE 3
Descriptive Statistics on the Variables

Variable	N	Mean	Std. Deviation	Minimum	Maximum
ln(EFFORT)	37	7.21	1.34	4.47	9.72
ln(QUAL)	31	2.77	0.95	0.69	4.56
ln(CTIME)	37	5.42	0.73	4.09	7.13
ln(SIZE)	37	4.43	1.83	1.81	8.00
ln(TEAM)	37	2.59	1.00	0	4.11
ln(COMPLX)	36	1.60	0.26	1.1	1.95
ln(PERCAP)	37	1.73	0.12	1.42	1.95
ln(REQUAL)	37	1.81	0.11	1.41	1.94
ln(REQVOL)	28	1.48	0.91	0	3.40
ln(MGROL)	37	0.74	0.9	-1.83	2.30
ln(INDEXP)	37	1.88	0.45	0.41	2.56
ln(SP)	37	-5.84	3.02	-12.21	-1.43
ln(MOD)	37	2.25	0.97	0	4.79

As seen in Table 3, the mean effort in the sample was $e^{7.21}$ or 1352.9 person-days. Correlations between independent variables are reported in Table 4. The number of observations used in calculating these correlations (28) is less than the total sample size (37) because of missing data for some variables in certain observations. It can be seen that there is only one large correlation (0.79), which is between project size and team size. Therefore, project size and team size were not used simultaneously as independent variables in the regressions. In our data set, 21 out of the 37 projects (that is, 57 percent) had team-estimated cycle time equal to the management-mandated cycle time. In these cases, the numerator in the expression to compute schedule pressure (SP) was set to 0.1 as described earlier.

TABLE 4
Correlations between Independent Variables $(N = 28)$

	ln(SIZE)	ln(TEAM)	ln(COMPLX)	ln(PERCAP)	ln(REQUAL)	ln(REQVOL)	ln(MGROL)	ln(INDEXP)	ln(SP)	ln(MOD)
ln(SIZE)	1	0.79*	0.32	0.17	0.19	-0.05	-0.02	0.17	0.26	0.21
ln(TEAM)		1	0.35	0.23	0.22	0.03	-0.27	0.2	0.18	0
ln(COMPLX)			1	0.44*	0.25	-0.42*	0.12	-0.02	-0.21	-0.02
ln(PERCAP)				1	0.49*	-0.21	0.09	0.24	-0.12	0.07
ln(REQUAL)					1	-0.21	0.16	0.05	-0.03	-0.19
ln(REQVOL)						1	-0.03	-0.24	-0.06	-0.07
ln(MGROL)							1	-0.23	0.07	0.22
ln(INDEXP)								1	0.28	0.19
ln(SP)									1	0
ln(MOD)										1

* Significant at 0.05

TABLE 5
Development Effort (EFFORT)

	With Backfiring	Without Backfiring
Intercept	4.49** (13.76)	4.56** (11.41)
ln(SIZE)	0.61** (9)	0.59** (7.46)
N	37	29
F	81.02**	55.62**
Adjusted R^2	0.69	0.66
* Sig. at 0.05; ** Sig. at 0.01		

TABLE 7
Cycle Time (CTIME)

	With Backfiring	Without Backfiring
Intercept	4.23** (17.89)	4.06** (14.65)
ln(Size)	0.27** (5.43)	0.29** (5.28)
N	37	29
F	29.51**	27.91**
Adjusted R^2	0.44	0.49
* Sig. at 0.05; ** Sig. at 0.01		

Regression results for effort, quality, and cycle time are reported below. We used forward stepwise regression because of the size of the data set. In each case, we added one variable from the list in Table 4. If a variable had a significant effect (at the 0.05 level) on the dependent variable (EFFORT, QUAL, or CTIME), we retained that variable in the regression model; otherwise, the variable was dropped. The numbers in Tables 5, 6, and 7 are the regression parameter estimates, and the numbers in parentheses are the t-statistics for the estimates. In all cases, we estimated a general linear model (GLM) including OrganizationID as a class (categorical) variable and found that there were no systematic differences between the organizations in terms of effort, quality, or cycle time. OrganizationID was insignificant at the 0.05 level; therefore, it was not used in the regression models.

The Shapiro-Wilk test was used to check the normality of residuals. The assumption of normality could not be rejected in any of the models at the 0.05 level of significance. White's test was used to check for the homoscedasticity of residuals, and in all cases, the assumption of homoscedasticity of residuals could not be rejected at the 0.05 level. Cook's distance was used to identify influential observations. The maximum values of Cook's distance were less than 0.2 in all models, indicating that there were no outliers. These checks indicated that there were no known major statistical problems in the regression models used.

4.1 Development Effort

Table 5 shows the factors that affected development effort. We found that project size was the only variable that significantly influenced development effort (see Appendix A for significance levels of all independent variables). There were no qualitative changes in the results when we removed the eight observations for which backfiring was used. Size explained more than 65 percent of the observed variance in effort in our sample. Overall sample results with backfiring data in Table 5 indicates that for each 1 percent increase in project size, effort increased by 0.61 percent.

TABLE 6
Quality (QUAL)

	With Backfiring	Without Backfiring
Intercept	1.38** (3.57)	1.11* (2.6)
ln(Size)	0.3** (3.87)	0.34** (4)
N	31	27
F	14.95**	16.7**
Adjusted R^2	0.32	0.38
* Sig. at 0.05; ** Sig. at 0.01		

4.2 Quality

Like development effort, quality was also affected by project size alone. Larger projects had more defects. Overall sample results in Table 6 indicate that for every 1 percent increase in project size, the number of defects increased by 0.3 percent. Size explained about 32 percent of the variance in quality. The sample size for quality was smaller than the sample size for effort because defect data was not available for six projects.

4.3 Cycle Time

With reference to cycle time, the results in Table 7 indicate that SIZE was the only significant predictor for cycle time. None of the other variables in Table 2 were significant in a model that also included SIZE. We found that an increase in SIZE increased CTIME. However, it may be noted that the increase in CTIME as a function of SIZE (0.27) was less than the increase in EFFORT (0.61). This was partly because larger projects generally had larger development teams (the correlation between SIZE and TEAM was 0.79), thereby reducing the impact of project size on cycle time.

4.4 Simultaneous Estimation of Models

The results presented above were based on models for effort, quality, and cycle time that were estimated separately. Prior research suggests that there could be dependencies among quality, effort, and cycle time; as a result, these equations should be estimated simultaneously. Further, as indicated earlier in Section 3.12, quality was expected to affect both effort and cycle time. We therefore also estimated the following system of equations using two-stage least squares:

$$\ln(\mathrm{QUAL}) = \beta_{10} + \beta_{11}\ln(\mathrm{SIZE}),$$
$$\ln(\mathrm{EFFORT}) = \beta_{20} + \beta_{21}\ln(\mathrm{SIZE}) + \beta_{22}\ln(\mathrm{QUAL}),$$
$$\ln(\mathrm{CTIME}) = \beta_{30} + \beta_{31}\ln(\mathrm{SIZE}) + \beta_{32}\ln(\mathrm{QUAL}).$$

Results are presented in Table 8. Comparing the results in Table 8 to the results in Tables 5, 6, and 7, we find that simultaneous estimation did not significantly affect the impact of SIZE on QUAL, EFFORT, or CTIME. The other independent factors continued to be insignificant. QUAL also did not influence EFFORT or CTIME in our sample.

TABLE 8
2SLS Estimation of Models

	QUAL		EFFORT		CTIME	
	With Backfiring	Without Backfiring	With Backfiring	Without Backfiring	With Backfiring	Without Backfiring
N	31	27	31	27	31	27
Intercept	1.38***	1.12*	3.8***	3.71***	4.12***	4.02***
Size	0.3***	0.34***	0.41*	0.3	0.2	0.25
Quality			0.6	0.8	0.16	0.09
R^2	0.34	0.4	0.62	0.59	0.42	0.5
R^2 adj	0.32	0.38	0.6	0.55	0.38	0.46
F	15***	16.7***	23***	17.2***	10.2***	12.3***
*** significant at 0.001; ** significant at 0.01; * significant at 0.05						

4.5 Verification of the Predictive Ability of Models

To test the reliability of our results and its usefulness in predicting software outcomes within and across organizations, we compared the predictions of our models to actual values from holdout samples. For within-organization comparison, for Organization A, which provided more than 50 percent of the observations in the sample, we had a holdout sample of five projects from Organization A. We compared the predictions of our model developed from the remaining projects of Organization A to the holdout sample for A (Section 4.5.1). For comparison across organizations, we estimated a model using data from organizations excluding Organization A and calculated the prediction errors for Organization A (Section 4.5.2). We also compared the predictions from the model based on the overall Organization A data set with the actual observations from other organizations (Section 4.5.3). In Section 4.5.4, we compared the predictions from the model based on overall sample data (excluding randomly selected observations for the holdout samples) with the holdout sample observations.

The accuracy of estimation was calculated as the mean magnitude of relative error (MMRE) percentage for N estimations as follows [15], [24]:

$$MMRE\% = \frac{\sum_{i=1}^{N} \left| \frac{Actual\ effort - Estimated\ effort}{Actual\ effort} \right| \times 100}{N}.$$

4.5.1 Organization A

We calculated MMRE by bootstrapping Organization A using 100 random bootstrapping samples. In each sample, five observations were retained as holdout observations, and the remaining observations were used for model estimation. The estimated model was used to calculate the errors for the five holdout samples. The means and standard deviations of MMREs for the 100 bootstrap samples are shown in Table 9. We found that the relative errors for effort and cycle time were very low, thereby indicating very high predictive accuracy, whereas the relative error for quality was relatively high.

TABLE 9
Mean and Standard Deviation of MMREs for Organization A

	Mean MMREs	Standard Deviation of MMREs
EFFORT	0.1042 (10.42%)	0.04
QUAL	0.2842 (28.42%)	0.16
CTIME	0.0888 (8.88%)	0.03

4.5.2 Prediction of Organization A Outcomes Based on Model Estimated from Other Organizations

In this case, all projects from Organization A were removed from the main sample and retained in the holdout sample. Regression parameters were estimated using project data from other organizations. Statistics on the relative errors for projects from Organization A are shown in Table 10. Note that in Table 10, MRE refers to the magnitude of relative error and is given by

$$\text{MRE} = |(\text{actual effort} - \text{estimated effort})/\text{actual effort}|.$$

Again, we found that the relative errors were low for effort and cycle time, but relatively high for quality.

4.5.3 Prediction of Other Organization Outcomes Based on Model Estimated from Organization A

To verify the portability of the model, we also checked to see whether a model estimated using observations from one organization (that is, Organization A) could satisfactorily predict observations for other organizations. We found that estimates for effort and cycle time had relative errors below 10 percent, which is still a low figure, whereas quality had a much higher relative error of about 49 percent with a high value for standard deviation as seen in Table 11, thereby making the model less portable for predicting quality.

TABLE 10
Mean and Standard Deviation of MREs for Organization A

	Mean MREs	Standard Deviation of MREs
EFFORT (N = 16)	0.1145 (11.45%)	0.08
QUAL (N = 11)	0.3341 (33.41%)	0.26
CTIME (N = 16)	0.1014 (10.14%)	0.67

TABLE 11
Mean and Standard Deviation of MREs for Other Organizations

	Mean MREs	Standard Deviation of MREs
EFFORT(N = 21)	0.0982 (9.82%)	0.06
QUAL (N = 20)	0.4944 (49.44%)	0.72
CTIME (N = 21)	0.0883 (8.83%)	0.08

TABLE 12
Mean and Standard Deviation of MMREs When Overall Data Set Is Used

	Mean MMREs	Standard Deviation of MMREs
EFFORT	0.089 (8.9%)	0.03
QUAL	0.2984 (29.84%)	0.19
CTIME	0.0818 (8.18%)	0.03

4.5.4 Overall Sample

MMREs for the overall data set using 100 bootstrap samples were calculated using a procedure similar to the procedure for Organization A. For each sample, its MMRE was calculated for five holdout observations. Means and standard deviations of the MMREs are shown in Table 12. Again, we could observe that the MMRE for effort and cycle time were superior to the MMRE for quality.

4.6 Benchmarks

The results from this study provide benchmarks for software development. These benchmarks are desirable since they are based on data collected from highly mature organizations that have achieved high control over variances during software development. The following equations are based on Tables 5, 6, and 7 and can be used as benchmark equations for effort, quality, and cycle time:

$$\ln(\text{EFFORT}) = 4.49 + 0.61^{*}\ln(\text{SIZE}) \quad (1)$$
$$\ln(\text{QUAL}) = 1.38 + 0.3^{*}\ln(\text{SIZE})$$
$$\ln(\text{CTIME}) = 4.23 + 0.27^{*}\ln(\text{SIZE}).$$

Various organizations can compare their software development outcomes to the benchmark outcomes obtained from the above equations. For example, if the development effort in any organization is greater than the effort predicted by the above effort equation for a given project, then, clearly, the organization's productivity is lower compared to the average productivity of efficient CMM level 5 organizations participating in this study. Similarly, benchmarks for cycle time and quality provide organizations with quantifiable goals to achieve in order to be efficient.

Lower values for the size exponent in the effort equation below (derived by taking antilogs of the benchmark effort in (1)) indicate higher economies of scale [19]:

$$\text{EFFORT}(\text{person-days}) = 89.12^{*}\text{Size}(\text{KSLOC})^{0.61}. \quad (2)$$

Equation (2) can be transformed to (3).

$$\text{Size}(\text{KSLOC}) = 0.000636^{*}\text{EFFORT}(\text{person-days})^{1.64}. \quad (3)$$

The equation with SIZE as the dependent variable indicates that as the applied effort increases, the volume of code written increases exponentially. By comparison, the exponent of size in the COCOMO II model for effort is 0.92 [7, p. 169]. In another study, Harter et al. [4, p. 462] have estimated the size exponent as 0.95. Using the data provided by Kemerer, we estimate the size exponent for Kemerer's data set as 0.81. It may be noted that all these estimates of the size exponent are higher than the exponent of our data sample (that is, 0.61). This indicates that the benchmarks presented in this paper correspond to relatively higher efficient software development processes, thereby indicating the benefits from reaching the highest level of CMM.

5 Conclusions

In this paper, we used a data set of 37 projects from four organizations that were at CMM level 5 to investigate the impact of various factors on software development outcomes. We found that software size was the most significant factor that affected development effort, quality, and cycle time.

The models, although parsimonious, achieved an MMRE of about 12 percent in predicting effort and cycle time and about 49 percent in predicting the number of defects in holdout samples. This compared extremely favorably to the widely used software estimation models that achieved MMREs in effort estimation ranging from 100 percent for FPs [15], [39] to 600-700 percent for COCOMO [15], [39]. In a study of data from the European Space Agency, the MMRE for the best effort estimation model was 36 percent [24].

Our results showed that the potential benefit of achieving high process maturity was a steep reduction in variance in effort, quality, and cycle time that led to relative uniformity in effort, cycle time, and quality. Our models for effort and cycle time appeared portable across organizations based on good predictions for effort and cycle time, whereas our model for quality appeared less portable. Our results were in contrast to the results of the European Agency study [24], which found that productivity differences among organizations were extremely important in estimating effort and that software estimation models were not portable across organizations.

Overall, our results indicated that the adoption of highly mature software development processes during software development reduced the significance of many factors such as personnel capability, requirements specifications, requirements volatility, and so forth. Discussions with our principal contacts at research sites indicated some reasons for the reduced significance of requirements-related factors: 1) increased adoption of best practices by client organizations that were to a great degree influenced by the software development organizations, thereby leading to well-defined requirements, and 2) software development organizations leveraging their expertise from prior engagements in assisting clients in requirements gathering and specification. Thus, most projects in our sample had high overall

TABLE 13
Significance of Independent Variables for Regressions in Sections 4.1, 4.2, and 4.3

Independent variable	Effort		Quality		Cycle-time	
	With Backfiring	Without Backfiring	With Backfiring	Without Backfiring	With backfiring	Without backfiring
ln(SIZE)	< 0.0001	< 0.0001	0.0006	0.0004	< 0.0001	< 0.0001
ln(COMPLX)	0.72	0.42	0.79	0.68	0.86	0.52
ln(PERCAP)	0.61	0.47	0.64	0.72	0.52	0.64
ln(REQUAL)	0.38	0.49	0.69	0.93	0.36	0.71
ln(REQVOL)	0.11	0.4	0.56	0.85	0.15	0.9
ln(MGROL)	0.95	0.77	0.59	0.32	0.55	0.59
ln(INDEXP)	0.43	0.63	0.35	0.5	0.38	0.72
ln(SP)	0.065	0.08	0.4	0.27	0.15	0.17
ln(MOD)	0.76	0.97	0.43	0.62	0.78	0.44

ratings for requirement specifications with minimum modifications (an average of only seven changes) after baseline specifications were agreed upon.

The results in this paper are constrained by certain limitations. First, the paucity of data imposed certain limitations on our statistical analysis. Most organizations treat application development capabilities and procedures to be proprietary and drivers of competitive advantage over competitors. As a result, it was difficult to obtain project data. Second, biases due to our convenience sample of projects imposed some limitation on the applicability of results. To reduce this bias, a random sample of projects from a random sample of CMM level 5 organizations would have been preferred. However, for reasons explained above, this was not possible. Third, the results were based on project data that were primarily MIS/business applications, thereby imposing some limitations on the generalizability to other application domains. Fourth, the use of backfiring for converting FPs to KSLOC was a possible source of data inaccuracies. However, these inaccuracies were limited since data analysis without including FP observations yielded similar results compared to the results from the data set that included FP observations. Finally, all our data was self-reported, leading to possible biases that provided a more positive view of the development process than what was really true. However, the use of tools and elaborate review procedures was likely to reduce this bias.

APPENDIX A

See Table 13.

ACKNOWLEDGMENTS

The authors are extremely grateful to all the staff members of organizations involved in this study for providing us with data. Without their participation, this study would not have been possible. The authors would also like to thank the three anonymous reviewers for their valued comments that helped in improving the quality of this paper. This paper also benefited from the comments of Rajiv Banker at a research workshop.

REFERENCES

[1] P. Jalote, *CMM in Practice: Processes for Executing Software Projects at Infosys.* Addison Wesley Longman, 2000.
[2] M.C. Paulk, "How ISO 9001 Compares with the CMM," *IEEE Software,* vol. 12, no. 1, pp. 74-83, Jan. 1995.
[3] T. Pyzdek, *The Six Sigma Handbook: The Complete Guide for Greenbelts, Blackbelts, and Managers at All Levels.* McGraw-Hill, 2003.
[4] D.E. Harter, M.S. Krishnan, and S.A. Slaughter, "Effects of Process Maturity on Quality, Cycle Time and Effort in Software Product Development," *Management Science,* vol. 46, pp. 451-466, 2000.
[5] H. Wohlwend and S. Rosenbaum, "Schlumberger's Software Improvement Program," *IEEE Trans. Software Eng.,* vol. 20, no. 11, pp. 833-839, Nov. 1994.
[6] M. Diaz and J. Sligo, "How Software Process Improvement Helped Motorola," *IEEE Software,* vol. 14, no. 5, pp. 75-81, Sept.-Oct., 1997.
[7] B.W. Boehm et al., *Software Cost Estimation with COCOMO II.* Prentice-Hall, 2000.
[8] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula, "Examining the Feasibility of a Case-Based Reasoning Model for Software-Effort Estimation," *MIS Quarterly,* vol. 16, pp. 155-171, 1992.
[9] D.E. Harter and S.A. Slaughter, "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis," *Management Science,* vol. 49, pp. 784-800, 2003.
[10] R.D. Banker, G.B. Davis, and S.A. Slaughter, "Software Development Practices, Software Complexity and Software Maintenance Performance: A Field Study," *Management Science,* vol. 44, pp. 433-450, 1998.
[11] G. Li and S. Rajagopalan, "Process Improvement, Quality and Learning Effects," *Management Science,* vol. 44, pp. 1517-1532, 1998.
[12] N. Ramasubbu et al., "Effect of Quality Management Practices in Distributed Offshore Software Development: An Empirical Analysis," *Proc. Academy of Management Meeting,* 2004.
[13] K. Chari and M. Agrawal, "Software Effort, Quality and Cycle Time: A Study," *Proc. INFORMS Conf. Information Systems and Technology,* 2005.
[14] A.J. Albrecht and J.E.J. Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.,* vol. 9, pp. 639-648, 1983.
[15] C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Comm. ACM,* vol. 30, pp. 416-429, 1987.
[16] J. Baik, "Disaggregating and Calibrating the Case Tool Variable in COCOMO II," *IEEE Trans. Software Eng.,* vol. 28, no. 6, pp. 1009-1022, Nov. 2002.
[17] "List of Published SCAMPI Appraisal Results," Software Eng. Inst., 2006.
[18] J.E.J. Gaffney, "Estimating the Number of Faults in Code," *IEEE Trans. Software Eng.,* vol. 10, no. 4, pp. 459-464, July 1984.
[19] R.D. Banker and C.F. Kemerer, "Scale Economies in New Software Development," *IEEE Trans. Software Eng.,* vol. 15, pp. 1199-1205, 1989.
[20] B.T. Compton and C. Withrow, "Prediction and Control of Ada Software Defects," *J. Systems and Software,* vol. 12, pp. 199-207, 1990.

[21] J.E. Matson, B.E. Barrett, and J.M. Mellichamp, "Software Development Cost Estimation Using Function Points," *IEEE Trans. Software Eng.,* vol. 20, pp. 275-287, 1994.
[22] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Trans. Software Eng.,* vol. 23, no. 11, pp. 736-743, Nov. 1997.
[23] M.S. Krishnan and M.I. Kellner, "Measuring Process Consistency: Implications Reducing Software Defects," *Management Science,* vol. 25, pp. 800-815, 1999.
[24] K. Maxwell, L.V. Wassenhove, and S. Dutta, "Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation," *Management Science,* vol. 45, pp. 787-803, 1999.
[25] B.K. Clark, "Quantifying the Effects of Process Improvement on Effort," *IEEE Software,* vol. 17, pp. 65-70, 2000.
[26] M.S. Krishnan et al., "An Empirical Analysis of Productivity and Quality in Software Products," *Management Science,* vol. 46, pp. 745-759, 2000.
[27] N. Nan, D.E. Harter, and T. Thomas, "The Impact of Schedule Pressure on Software Development: A Behavioral Perspective," *Proc. Int'l Conf. Information Systems,* 2003.
[28] P.C. Pendharkar, G.H. Subramanian, and J.A. Rodger, "A Probabilistic Model for Predicting Software Development Effort," *IEEE Trans. Software Eng.,* vol. 31, pp. 615-624, 2005.
[29] "True S and Price S: Software Development and Lifecycle Estimating Models," PRICE Systems, 2006.
[30] "CA-Estimacs," Computer Assoc., 2006.
[31] "SEER-SEM," GA SEER Technologies, 2006.
[32] B.W. Boehm, *Software Engineering Economics.* Prentice-Hall, 1981.
[33] *Function Point Counting Practices Manual.* Int'l Function Point Users Group, 2006.
[34] S.N. Mohanty, "Software Cost Estimation: Present and Future," *Software—Practice and Experience,* vol. 11, pp. 103-121, 1981.
[35] C.A. Behrens, "Measuring the Productivity of Computer Systems Development Activities with Function Points," *IEEE Trans. Software Eng.,* vol. 9, no. 6, pp. 648-652, Nov. 1983.
[36] H.A. Rubin, "Macroestimation of Software Development Parameters: The Estimacs System," *Proc. SOFTFAIR Conf. Software Development Tools, Techniques, and Alternatives,* 1983.
[37] R.D. Banker and S.A. Slaughter, "The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement," *Information Systems Research,* vol. 11, pp. 219-240, 2000.
[38] D.R. Goldenson and D.L. Gibson, "Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results," Technical Report CMU/SEI-2003-SR-009, Software Eng. Inst., 2003.
[39] S.S. Vicinanza, T. Mukhopadhyay, and M.J. Prietula, "Software-Effort Estimation: An Exploratory Study of Expert Performance," *Information Systems Research,* vol. 2, pp. 243-262, 1991.
[40] T. Mukhopadhyay and S. Kekre, "Software Effort Models for Early Estimation of Process Control Applications," *IEEE Trans. Software Eng.,* vol. 18, no. 10, pp. 915-924, Oct. 1992.
[41] C.K. Prahalad and M.S. Krishnan, "The New Meaning of Quality in the Information Age," *Harvard Business Rev.,* vol. 1999, pp. 109-118, 1999.
[42] ISO/IEC 9126-1, 2001, Int'l Standards Organization, 1991.
[43] C. Fox and W. Frakes, "The Quality Approach: Is It Delivering?" *Comm. ACM,* vol. 40, pp. 25-29, 1997.
[44] D.E. Harter and S.A. Slaughter, "The Cascading Effect of Process Maturity on Software Quality," *Proc. Int'l Conf. Information Systems,* 2000.
[45] R.D. Austin, "The Effects of Time Pressure on Quality in Software Development: An Agency Model," *Information Systems Research,* vol. 12, pp. 195-207, 2001.
[46] V. Basili, "The Experience Factory and Its Relationship to Other Quality Approaches," *Advances in Computers,* vol. 41, pp. 65-82, 1995.
[47] W.S. Humphrey, "Characterizing the Software Process: A Maturity Framework," *IEEE Software,* vol. 5, no. 3, pp. 73-79, Mar. 1988.
[48] M.C. Paulk et al., "Capability Maturity Model, Version 1.1," *IEEE Software,* vol. 10, no. 4, pp. 18-27, July 1993.
[49] F.P.J. Brooks, *The Mythical Man-Month,* second ed. Addison-Wesley, 1995.
[50] M. van Genuchten, "Why Is Software Late? An Empirical Study of Reasons for Delay in Software Development," *IEEE Trans. Software Eng.,* vol. 17, no. 6, pp. 582-590, June 1991.
[51] D.E. Harter, S.A. Slaughter, and M.S. Krishnan, "Benefits of CMM-Based Process Improvements for Support Activities—An Empirical Study," *Proc. Am. Conf. Information Systems,* 1998.
[52] M.J. Benner and M. Tushman, "Process Management and Technological Innovation: A Longitudinal Study of the Photography and Paint Industries," *Administrative Science Quarterly,* vol. 47, pp. 676-706, 2002.
[53] L.H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Trans. Software Eng.,* vol. 4, no. 4, pp. 345-361, July 1978.
[54] R.D. Banker, S.M. Datar, and C.F. Kemerer, "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects," *Management Science,* vol. 37, pp. 1-18, 1991.
[55] R. Nelson, "Educational Needs as Perceived by IS and End-User Personnel: A Survey of Knowledge and Skill Requirements," *MIS Quarterly,* vol. 15, pp. 503-525, 1991.
[56] IEEE Std. 830-1998, *IEEE Recommended Practice for Software Requirements Specifications,* IEEE, 25 June 1998.
[57] E.B. Swanson and E. Dans, "System Life Expectancy and the Maintenance Effort: Exploring Their Equilibrium," *MIS Quarterly,* vol. 24, pp. 277-297, 2000.

Manish Agrawal received the BTech and MTech degrees in electrical engineering from the Indian Institute of Technology at Kanpur, India, and the PhD degree in management information systems from State University of New York, Buffalo, New York. He is an assistant professor in the Department of Information Systems and Decision Sciences, University of South Florida, Tampa. His research interests include software engineering, fraud detection, and electronic markets. His research has received the Best Paper Award at the America's Conference of Information Systems (AMCIS). He is a member of Association for Information Systems (AIS) and the Institute for Operations Research and the Management Sciences (INFORMS).

Kaushal Chari received the BTech degree in mechanical engineering from the Indian Institute of Technology, Kanpur, India, in 1984, and the MBA and PhD degrees with specialization in management information systems from the University of Iowa in 1986 and 1990, respectively. He is currently the chair and an associate professor of information systems and decision sciences at the University of South Florida, Tampa. His research interests include decision support systems, software engineering, software agent applications, distributed systems, and fraud detection. He currently serves as the associate editor of management information systems for *Interfaces* journal and the secretary/treasurer of the Institute for Operations Research and the Management Sciences (INFORMS) Information Systems Society. He is a member of the Association for Information Systems (AIS) and INFORMS.