THE DETERMINANTS OF QUALITY OF SOFTWARE DEVELOPMENT PROJECTS

Kallol Bagchi, University of Texas at El Paso (UTEP), kbagchi@utep.edu, 915 747 5376
Somnath Mukhopadhyay, UTEP, smukhopadhyay@utep.edu, 915 747 7720

ABSTRACT

Empirical results on the technical antecedents of quality of software development projects are rare. Examining 592 cross-national projects, this research finds that quality is influenced by team size, method used and underlying architecture. The difference in antecedents of quality between projects implemented in earlier periods (before 1999) and in later periods (in or after 1999) is significant: the quality of early year projects depends on team size, data quality, method and programming language used; the quality of later year projects depends on team size, method used and CASE tools. Results were obtained using a SEM analysis in AMOS.

Keywords: Software Development Projects, Software Defects, Structural Equation Modeling, Cross-national Study, Software Quality

INTRODUCTION

Many software projects are late, over budget, unpredictable and sometimes fail even before delivery (Reel, 1999; Brooks, 1995). Cusumano et al., (2005) pointed out that the global software product business is in serious trouble with falling prices, piracy and availability of good quality free open source software. This observation points to the need of producing quality software, software that is free from errors. Although quality has many facets, we define the quality of a software as one which is measured by the number of defects—the lesser the number of defects in the software, the better the quality of the software is. The present empirical study is on determining technical factors of development software quality. Empirical results on software quality involving development software projects from multiple nations are rare. This study attempts to fill that gap.

The unique features of the study are:

• Analysis of a large number of software development projects (> 500 in number)
• Projects from a multiple number of nations (> 20 in total)
• Projects covered over a large period of time (1991-2006)
• Focus on project technical features and investigation of roles of national-level indicators
BACKGROUND AND MODEL

Software is embedded in many activities of day-to-day modern society and every year billions of dollars are lost in the US alone in software glitches. “Software Hall of Shame” documents many IT projects that have failed (Charette, 2005). It can cause human deaths in some situations (Software Bug, 2008). Software failure happens in all nations, all firms and have been with the software development process since its beginning. A prevention of failure strategy is necessary but not many firms pay careful attention to it, probably due to limitations of resources and cost- and time-constraints involved, among other things. Cost of fixing errors arguably could be 100 times more if it is detected when implemented than when detected during the development stage. Many factors contribute to a software error and thus quality. Some of these are (Charette, 2005; Krishnan et al., 2000): lack of better processes and lack of investments in the initial stages, badly defined system requirements, unrealistic project goals, lack of personal capability, poor communication among stakeholders, use of immature technology, poor development practices etc. to name a few. Standish group reports that the success rate of software development project rate has slowly gone up from 16% in 1994 to 31% in 2003, at a linear rate of only 1.7% a year (Marasco, 2006).

The team size of a project may contribute to number of errors/defects as also the size (Brooks, 1995). Similarly, the effort spent can improve the quality. A project’s technical components—hardware and software as well as development practices adopted could be crucial to the quality of the project. A progress in programming style, techniques and development methodologies can help programmers in developing software with less errors (Huizinga and Kolawa, 2007; McDonald, Musson and Smith, 2007). On the other hand, the use of latest technology (in many cases immature and untested) is enticing but may not be always prudent as this can lead to failures (Charette, 1995). The present study takes the lead from this observation and tries to find out the technical factors that could be responsible for software errors with empirical testing of such factors. In particular, the focus of the study is on technical characteristics of the project—programming languages, tools, methods, architectures used (to name a few) in order to find empirical evidence on how these elements contribute to errors and thus in the success of a software development project.

The Drivers of Quality

We first describe the drivers of quality as follows:

- **The Data Quality**: According to Wang and Strong (1996), more than 60% of firms surveyed in their study had problems with data quality. Accuracy, completeness, conciseness, easy interpretability and accessibility of data are some factors needed to ensure data quality. The better the data, the lesser is the number of errors (Ballou and Pazer, 1985; Umble et al., 2003; White et al., 2002).

- **The Architecture (Client-server or Not)**: Client-server architecture can reduce the number of errors. A client-server software architecture may give a better understanding of the software structure; in particular, it makes the software system’s property and its
developmental process clearer by modular design. By breaking up the software into components on both client and server sides, one can keep the design clean and thus reduce errors in design. (Gomaa, 1997). Material printed in its entirety in the Institute’s Proceedings is considered published. The copyrights for all forms of presentation at the Institute’s Annual Meeting remain with the authors.

- **The Programming Language Used (Generation):** Later generation languages can introduce more complexity in coding and so more errors in the software design. On the other hand, languages such as C++, Java, C# are sophisticated and have libraries that can be reused which may reduce error (Marasco, 2006).

- **The Tools Used (CASE or Otherwise):** Methodology and tools are the two areas where most money has been spent in the past 25 years, according to Marasco (2006). Each year Fortune 1000 firms spends $3 million on software development tools and gets less than a 3% point improvement in project success rates. Many of the tools actually may save time and eliminate errors. Introducing CASE tools in the design cycle may actually decrease the errors in the software design. Recently, configuration management and version control systems in ERP development have improved a lot that may help software developers. CASE tools many times provide help for completeness and consistency checking and assist developers to produce complete and correct specifications and systems per user requirements thus reducing errors (Flynn et al., 1995).

- **The Design Method Used:** The development methods have evolved significantly: from structured design to object-oriented to agile methods in recent times. Standards such as Unified Modeling Language (UML) and Rational Unified Process have also helped in software project design (Marasco, 2006). It is now accepted that the use of a methodology can reduce the errors in the software design and thus improve the quality. The US Software Engineering Institute, Carnegie Mellon University has developed the capability maturity model (CMM) which has multiple levels of perfection. Level 3 of CMM is the one at which a firm has characterized its practices and claim to understand it. About only 30% of all US firms reported that they were at level 3 of CMM practices (Charette, 1995). A rigorous development approach may help to design better software.

- **The Size of the Project:** The larger the size, more is the number of errors and less is the quality (Martin et al., 2005). Large-scale projects fail 3-5 times more than smaller ones (Charette, 1995). Both static and interactive parts of the project get complicated with the increase of the project size. Size is the single most important factor in quality at CMM level 5 (Ramachandran, 2008).

- **The Team Size of the Project:** Brooks’ law states that merely adding manpower to a late software project gets it delayed (Brooks, 1995). The larger the team size, more could be the error and less is the quality.

- **The Effort Expanded:** The larger the effort, the less is the error and better is the quality. We also enquire about the drivers of size, team size and effort as these contribute to overall quality of a project.

**The Drivers of Size**

The Drivers of size are described next.
• **The Data Quality**: The data quality may be related to the size of the project: the higher the data quality, the lesser is the size (Charette, 1995).

• **The Design Method Used**: Size could be dependent on the design scheme used. Size could be larger if no design scheme is used.

### The Drivers of Effort

The Drivers of Effort are listed below:

• **The architecture (Client-server or not)**. As observed earlier, Client-server architecture can reduce the effort.

• **The language used (generation)**. Later generation languages can introduce more complexity in coding and so more effort in the software design.

• **The tools used (CASE or otherwise)**. CASE tools are not easy to use and learn. Introducing CASE tools in the design cycle may actually increase the effort in the software design. Diagramming tools and the information repository may not always increase efficiency and thus productivity. On the other hand, prototyping tools and code generators may lead to better efficiency and thus productivity (Flynn et al., 1995).

• **The design method used**. Use of a methodology can reduce the effort in the software design.

• **The size of the project**. Size is the single most important factor in effort at CMM level 5 (Ramachandran, 2008). The larger the size, the more is the effort.

• **The Team size of the project**. According to Blackburn et al., (2008), factors responsible for increasing the effort put in a development project include increased communication, integration tests, and rework. A large team means that the project is broken up into smaller parts or modules, increasing the number of communication links, and thereby increasing the effort (Brookes 1995; Hoedemaker et al., 1999). So, the larger the team size, more is the effort.

### The Driver of Team Size

Only one factor is considered here.

• **The Size of the Project**. The larger the size, the larger is the team size. Complexity of a project and thus size increases the team size of a project (Blackburn et al., 1996; Blackburn et al., 2008). The model is shown in Figure 1.
RESEARCH QUESTIONS

The first research question follows from the above discussion.

- **Research Question 1 (RQ1).** What technical factors drive the quality, effort and size of software projects at a global level?

Since these development projects come from multiple years, the role of year in which the developmental software project got completed may have an influence on quality and so is also investigated. It is known that more errors could have occurred in software produced in early years; later years may show production of less-error ridden software than others because of the advances in tools and techniques in software design and an increased awareness of the importance of error-free software systems consisting of code and data. Early year software projects additionally could have the possibility of developing more errors due to lack of tool support, interpersonal communications needed between people engaged in designing such software and lack of latest tools used in communications. One of the major incidents in raising awareness in error-free software systems was the “Year 2000” or Y2K problem. The Y2K (or the year 2000) problem raised the awareness of developing error-free codes. According to http://en.wikipedia.org/wiki/Y2K (Wikipedia, 2006):

“The Year 2000 problem (also known as the Y2K problem, the millennium bug, the Y2K bug, or simply Y2K) was the result of a practice in early computer program design that caused some date-related processing to operate incorrectly for dates and times on and after between December 31, 1999 and January 1, 2000 and on other critical dates which were billed "event horizons". This fear was fueled by the attendant press coverage and other media speculation, as well as corporate and government reports. People recognized that long-working systems could break down when the "...97, 98, 99..." ascending numbering assumption suddenly became invalid. Companies and organizations world-wide checked and upgraded their computer systems.

The total cost of the Y2K was estimated at over 300 billion US dollars. Although it was a data related coding problem, many experts cited benefits and awareness that may have resulted from the Y2K issue in preparing better quality software systems later on.

- **Research Question 2 (RQ2).** Do early-year (typically pre Y2K) projects and later-year (typically post Y2K) projects have similar characteristics that contribute to quality?
The Release 10 database of the Data Repository of the International Software Benchmarking Standard Group (ISBSG), was used which has 4106 projects in the Repository (ISBSG, 2007). Project data submission was voluntary and many projects do not have data for all the data fields that ISBSG offers. The project repository data have come from twenty countries, with 70% of the projects being less than six years old. A broad range of project types from many industries and many business areas are available. The submitted data are voluntary and have been validated in previous studies (Jeffrey et al., 2000; Lokan, 2000). The project data has been used in academic studies (Heales et al., 2004) and in many industrial settings for estimation and benchmarking purposes. Of the 4106 projects from the global project archive, due to missing data, only 592 qualified as viable projects in the present study that contained data for relevant variables (especially dependent variable) considered in this study. A SEM model was developed.
in AMOS and used for answering RQ1 (Bollen, 1989). Of the 592 data points considered, missing values of several indicator variables were treated by AMOS with full interpretation MLE scheme (Arbuckle, 2006). This, according to many researchers, is a better method of treating missing data. For answering RQ2, a multi-group SEM approach was taken to assess the impact of the age of a project on the number of defects (AMOS, 2010).

<table>
<thead>
<tr>
<th>Paths</th>
<th>Estimate</th>
<th>Standard Error (S.E.)</th>
<th>Critical Ratio (C.R.)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Determinants of Size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size ← DataQ</td>
<td>-123.07985</td>
<td>106.94802</td>
<td>-1.15084</td>
<td>0.24980</td>
</tr>
<tr>
<td>Size ← E1</td>
<td>1610.99420</td>
<td>47.67324</td>
<td>33.79243</td>
<td>0.00000</td>
</tr>
<tr>
<td>Size ← Method</td>
<td>-1315.59960</td>
<td>202.74474</td>
<td>-6.48895</td>
<td>0.00000</td>
</tr>
<tr>
<td><strong>Determinants of Team Size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team Size ← E1</td>
<td>7.35121</td>
<td>0.40088</td>
<td>18.33772</td>
<td>0.00000</td>
</tr>
<tr>
<td>Team Size ← Size</td>
<td>0.00396</td>
<td>0.00029</td>
<td>13.69851</td>
<td>0.00000</td>
</tr>
<tr>
<td><strong>Determinants of Effort</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effort ← APL (Adv. Prog. Language)</td>
<td>-947.55759</td>
<td>871.28822</td>
<td>-1.08754</td>
<td>0.27680</td>
</tr>
<tr>
<td>Effort ← Size</td>
<td>-3.41139</td>
<td>0.54640</td>
<td>-6.24337</td>
<td>0.00000</td>
</tr>
<tr>
<td>Effort ← Architecture</td>
<td>1479.40970</td>
<td>1277.46300</td>
<td>1.15808</td>
<td>0.24683</td>
</tr>
<tr>
<td>Effort ← E2</td>
<td>9246.85630</td>
<td>549.76036</td>
<td>16.81980</td>
<td>0.00000</td>
</tr>
<tr>
<td>Effort ← Method</td>
<td>-2630.25170</td>
<td>1780.14560</td>
<td>-1.47755</td>
<td>0.13953</td>
</tr>
<tr>
<td>Effort ← CASE</td>
<td>2978.51640</td>
<td>1417.54090</td>
<td>2.10119</td>
<td>0.03562</td>
</tr>
</tbody>
</table>
### RESULTS AND DISCUSSIONS

Preliminary results for answering RQ1 shows that (Table 1):

- A lack of methods affects large sized projects
- Effort increases with large sized projects and decreases when method is used; the use of CASE tools has impact on effort. Large team size increases effort
- Quality depends on the team size of the project and technical features such as use of methods and the underlying architecture.
- The size of the coefficients of errors suggests that other significant indicators may be considered

The results show that the overall SEM model is a good fit (RMSEA is 0.01<0.5, various fit measure values are above 0.9). This answers RQ1.
For answering RQ2, first it was tested whether Before_1999 and After_1999 project groups exhibit the same characteristics. For this, a multi-group analysis of the data set using Before_1999 and After_1999 classifications of project groups was done. Statistical tests lead to the conclusion that Before_1999 and After_1999 project models were statistically reasonable. $R^2$ values are given in vary from 0.38 (for later years) to 0.50 (for earlier years). Both models, Before_1999 and After_1999, satisfactorily fit the data. The RMSEA values are $<$0.04 and $<$0.10 for the two year groups. Additionally the values of fit indices are all $>$0.90. Next the similarity of Before_1999 and After_1999 models was checked. Examining non-standardized coefficients of two models, it was observed that the models are indeed different. For example, the impact of team size and method on quality has increased in later years (as evidenced by larger coefficient values). Examining some of the path coefficients, it was also observed that impact of team size and effort influences quality in both periods (before and after). Quality of later year projects additionally depends on size, although the size changes sign. This requires additional investigation. Technical factors were significant for quality, but not uniformly for all models; for the overall model, method and architecture proved to be significant. Quality of earlier year projects depends on data quality (strongly), APL, and methods (weakly, 1-t), whereas quality of later year projects depends on CASE tools (strongly) and methods (weakly, 1-t).

The concern for data quality to ensure project quality was in existence in earlier period. Method use and use of later generation programming languages reduced the number of defects in this period. The concern for project quality was reflected in later period by increased use of CASE tools and methods that can reduce software defects. This concern may have resulted from Y2K awareness, although there could also be other reasons.

**CONCLUSIONS**

This preliminary research adds to the literature on the topic of software project quality. Empirical results on the quality of global development software projects are rare. Several technical antecedents statistically contribute to quality; in particular, team size is extremely influential to quality for all the models introduced in the article. The main drivers of effort were size of the project and team size. Also size of the project influenced team size as expected. Size has been found to be the single most important indicator of quality for CMM level 5 projects in previous empirical studies. The empirical results of the present paper show that team size, is also very important for all project types. It is possible that size may be indirectly exerting influence on quality through team size. A technical factor such as method was also responsible for the overall quality of the project for all models. Other technical factors also proved relevant for various models considered.

The multi-group analysis was based on non-equal sizes of data driving the two models, so results need to be interpreted with caution. For the period grouping, although the multi-group model provided statistically inferior fit than the single model, yet the multi-group models of provided...
statistically acceptable fits (based on RMSEA and several fit statistics). Presently, research is going on to include other types of variables (Gopal and Koka, 2009). We for example, found indications that both for inhouse and offshored software projects, culture plays a significant role on software quality (Hofstede, 2001).

References

AMOS (2010). FAQ #3: Multiple group analysis, Web site of Division of Statistics and Scientific Computation, College of Natural Sciences, The University of Texas at Austin.


