Conformance Quality and Failure Costs in the Software Industry: An Empirical Analysis of Open Source Software

Lars M. Karg[1,2], Michael Grottke[2], Arne Beckhaus[1]
[1]SAP Research, Darmstadt, Germany
[2]University of Erlangen-Nuremberg, Nürnberg, Germany

Abstract – **The quality cost concept is well known in production economics. Recently, it has received a lot of attention in the field of software engineering. However, empirical studies of the association between failure costs and conformance quality have only been conducted for closed source software projects, but not for open source projects. This paper addresses this research gap. On the one hand, our analysis revalidates findings from production economics. On the other hand, it extends the limited empirical knowledge in the software quality cost research domain.**

Keywords – **Conformance quality, empirical analysis, failure costs, open source software, production economics**

I. INTRODUCTION

For decades, users of software solutions have been suffering from poor solution quality [1]. Despite the tremendous effort spent on software quality improvement, more than half of all faults (also referred to as defects) are still not found during testing, but after shipment [2]. A 2002 study concludes that software faults not found due to an inadequate testing infrastructure account for annual economic damages of 38 billion dollars in the U.S. alone [3]. High complexity and tight development schedules are often seen as the main reasons for the large percentage of faults remaining in the released software product [4].

New software engineering approaches may help to overcome the quality challenge. Open source software (OSS) development is one of the approaches that have recently gained significance. It is receiving more and more interest in the research community; some researchers even consider it the next big software development paradigm [5]. However, the OSS development approach is still not fully understood. Some aspects, such as the economics of OSS development [6] or learning in OSS projects [7], have already been addressed sufficiently. Others, such as quality costs in OSS projects, have received less attention.

The goal of this paper is to address this research gap. We study the association between conformance quality (i.e., the conformance of a software product with its requirements) and failure costs (i.e., the quality costs caused by failure report processing and fault removal after the software has been released) in OSS projects. By this, we do not only revalidate findings from closed source projects and production economics, but we also extend the limited knowledge in the software quality cost research domain.

The remainder of this paper is structured as follows: In Section II, we develop our research framework and hypothesis. Section III presents the research site, the data collection procedure, and the model estimates. We interpret the results in Section IV and close the paper with Section V, providing directions for further research.

II. CONFORMANCE QUALITY & FAILURE COSTS

A. Theoretical Background

The cost impact of quality was first realized during the 1930s in industrial engineering [8]. Since then, the quality cost concept has been adopted by many engineering disciplines. In the 1980s, software engineers began conducting research on the concept, but the challenge of adjusting it to the characteristics of software development is still being faced [9]. Consistent with definitions from industrial engineering, software quality costs are defined as the costs "incurred in the pursuit of [software conformance] quality or in performing quality-related activities" [10], p. 196. According to cost accounting, quality costs can further be structured by different classification schemes. If any is used at all, the PAF (prevention, appraisal, and failure) cost scheme seems to be the one most commonly applied to software development [10, 11]. It distinguishes between three activity types (and corresponding quality cost categories) [10]:

- Prevention activities, such as quality planning and training;
- appraisal activities, such as testing, control, and measurement; and
- failure-related activities, such as rework, failure mode analysis, and corrective maintenance. (Note that we do not distinguish between internal and external failure costs.)

While the concept of quality costs is well known in the field of software engineering, only few empirical studies investigating the association between conformance quality and software quality costs can be found in the literature[12, 13]. This is in contrast to industrial engineering, where empirical studies are regularly published [14]. One reason for the small number of software-related studies might be the limited availability of software quality cost data in

companies, as well as the fact that researchers often do not obtain access to those data that companies collect. Moreover, to be widely applicable in software development, the PAF classification scheme needs further adaptation [11]. While for most other engineering disciplines at least a basic discipline-adjusted quality cost scheme is available, there is no such tailored scheme for software engineering [15].

It is widely accepted by research and practice that the largest part of the quality costs in software development falls into the failure costs category. Therefore, these costs are of special interest in the continuing debate of software quality costs [9, 16]. Nevertheless, empirically grounded understanding of factors influencing failure costs is still limited [12, 13].

In software development, failure costs are mostly driven by the effort spent on processing failure reports and removing the underlying faults in the source code. Other failure-related costs, such as those that typically occur for manufactured products, are of less cost concern, because software is a digital product: In contrast to physical products, most of its parts can be changed after development with relatively little effort [17]. In addition, costs due to damages at the user-side account for (external) failure costs; however, they are often hidden and thus cannot easily be measured. Therefore, studies often exclude them or estimate them by rule of thumb [18]. In the following, we thus understand software failure costs as the costs associated with the processing of failure reports and the fixing of faults.

B. Research Framework

According to the previously developed understanding, we model failure costs (Costs) as a function of the delivered software conformance quality (Quality) as well as several control variables (see Fig. 1):

$$\text{Costs} = f(\text{Quality}, \text{control variables})$$

Literature suggests a strong negative association between conformance quality and failure costs [12, 13, 19, 20]. With improving conformance quality, the number of faults in a software product decreases; assuming that at a lower fault density the complexity and virulence of the existing faults is not higher, this implies a reduction in failure costs. In accordance with prior research, we therefore formulate the following hypothesis:

Higher conformance quality is associated with lower software failure costs.

Our model controls for several important influences. Prior empirical research has suggested that product size has a significant influence on developer performance [21]: As a software product grows in size, its complexity increases dramatically, making it more difficult for a developer to understand its dependencies.

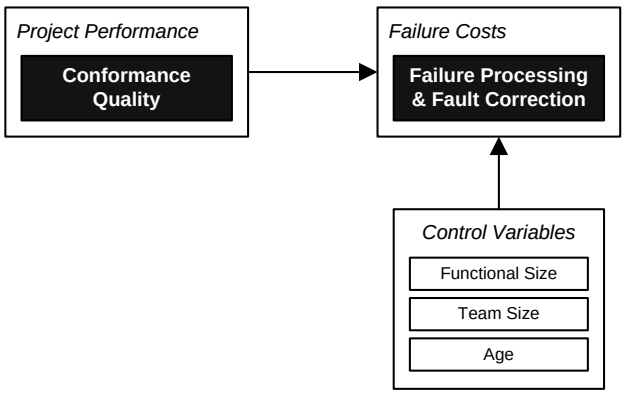

Fig. 1. Conceptual Framework.

This effect increases the necessary effort to comprehend the cause of a software failure and to fix the related fault.

One effect of the collaborative software development process can be observed at team level. Previous studies have shown that software development performance can be negatively associated with team size and its dispersion [22]: Larger teams tend to require higher coordination effort, which increases even more strongly if development is distributed or – in the case of OSS – virtually distributed. It seems reasonable to expect such an effect for the processing of failures and the fixing of faults as well.

The last control variable in our model is the age of the project: It can be assumed that over time experience on how to process failure reports and to fix faults effectively is accumulated in a project [21].

III. METHODOLOGY

A. Research Site and Data Collection

As most prior empirical studies on OSS development projects, we collected the data from SourceForge.net. At the end of March 2009 (the time we conducted our study), SourceForge.net hosted more than 140,000 OSS projects, thus being the world's largest repository. Besides mere storage, SourceForge.net offers a useful set of services to manage the OSS development process, including bug tracking and mailing lists. Many of the hosted projects use these free services and thus store data of their development process on the SourceForge.net servers. In line with the open source philosophy of the hosted software sources and products, these process data are made available to the research community. The large sampling population and the wealth of available data per project make SourceForge.net the ideal site to collect data for research on the OSS development process [23].

For the purpose of investigating the association between conformance quality and failure costs, we primarily rely on the data related to three SourceForge.net services:

1. the repository itself (to access the source code of each considered project);
2. the bug tracker (to gather basic information about each reported failure, such as processing time);
3. and the general project statistics (to collect project statistics such as team size).

Not all projects hosted at SourceForge.net are suitable for our study: Some do not make use of the bug tracker, which prevents us from calculating processing times, while others are in alpha state and have not yet released a stable version of their software source and product. In addition, like all empirical studies our study depends on the comparability of the objects under investigation. In order to ensure comparability, we followed a strict selection process.

First, we manually selected and extracted the first 250 OSS projects listed under the non-exclusive categories 'Enterprise' and 'Financial' in the SourceForge.net software map. After removing duplicates classified under both categories, our set contained 483 unique projects.

Second, we ranked these 483 projects by two criteria: development activity, which is a metric calculated by SourceForge.net expressing how active the project is, and the number of downloads, another metric provided by SourceForge.net. Based on these two criteria, we removed 129 projects of low activity or usage, resulting in a set of 354 projects.

Finally, we ensured that all projects have a development status of at least 5 indicating high process and product maturity, that at least 5 developers are registered with the project, and that the bug tracker is actively used. The first two criteria were verified based on measures provided by SourceForge.net; for the third one we manually checked the bug tracker.

Following this three-step approach, we composed a final set of 32 comparable projects suitable for our study.

B. Variable Measurement

The **conformance quality** (Quality) of a software product is measured as the software size in lines of code (LOC) divided by the number of failures reported by users in the bug tracking system. This measure is similar to the ones used in previous studies [12, 22]. Since different programming languages are employed across projects, we use the SLOCCount tool to guarantee a consistent measure (see http://www.dwheeler.com/sloccount).

As discussed in the previous section, in software development most of the **failure costs** (Costs) are caused by failure report processing and fault removal. Thus, the effort spent on these activities seems to be a good proxy for failure costs. However, for OSS projects' failure processing effort is hardly ever reported. As an example, consider the well-known and professionally managed JBoss project: For only about 6% of all failure reports, effort data are available [24]. Even worse, no effort data at all are provided for projects hosted on SourceForge.net. In consequence, we were not able to use the actual processing effort. We instead relied on the processing time inferred from the bug tracker as a proxy for failure costs: We calculated the processing time of a single failure as the time span (in minutes) between the initial report and its final closure, and the total failure costs of a project as the sum of the processing times of all reported failures. Of course, this metric can only be expressive if the bug tracker is actively used–our selection process (described in the last section) has helped to ensure this property.

As mentioned above, our model also includes a number of control variables. Since different OSS projects utilize different programming languages, we could not simply use the LOC to control for **functional size** (FuncSize). For each project, we therefore used the conversion table provided in [25] to convert the LOC measured with the SLOC Count tool into function points . This approach resulted in functional size measures comparable across the 32 projects.

Team size (TeamSize) is given by the number of people involved in the project according to the SourceForge.net project statistics. This measure is derived from explicit registrations with a project and therefore represents the core team, while the periphery of occasional bug reporters is ignored [22].

Age measures the number of minutes since a project was registered at SourceForge.net. This variable helps to control for virtual organizational learning effects [7].

C. Empirical Analysis and Results

Prior empirical research on software engineering economics recommends a log-linear model of project performance [21, 22]. The log transformation ensures that all estimated cost values are positive. Additionally, the fact that economics of scales have been observed suggests that a log-linear model may indeed be adequate [21, 22]. We therefore propose the following model:

$$\ln(\text{Costs}) = \beta_0 + \beta_1 \cdot \ln(\text{Quality}) + \beta_2 \cdot \ln(\text{FuncSize}) + \beta_3 \cdot \ln(\text{TeamSize}) + \beta_4 \cdot \ln(\text{Age}) + \varepsilon.$$

All of the following analyses were conducted with the statistical software package R [26].

We first used Ramsey's RESET [27] to test the adequacy of our model; no misspecification was detected. We then estimated the model parameters via the ordinary least squares (OLS) approach. Next, we performed several specification checks for the estimated model to ensure that the OLS assumptions are satisfied. The visual examination of the residual plot and the DFFIT measure by Welsch and Kuh indicated no influential observation.

We checked for the presence of multicollinearity using variance inflation analysis. Variation inflation factors (VIF) above 5.3 indicate possible problems due to multicollinearity [28].

TABLE I
OLS REGRESSION RESULTS

Variable	Estimate	*t* value
(Intercept)	5.5183 (3.2986)	1.673
Conformance Quality	-0.83162 (0.14173)	-5.867***
Functional Size	0.79858 (0.21150)	3.776***
Team Size	0.05469 (0.12346)	0.443
Age	0.64388 (0.30948)	2.081*
R-squared: 0.6557 Adj. R-squared: 0.6047		
F statistic: 12.8532 *p* value < 0.0001		

Note: *: significant at 5%; ***: significant at 0.1%;
Standard errors are given in brackets.

For our explanatory variables, the VIF values ranged from 1.41 to 3.82, with a mean VIF of 2.37; we therefore concluded that multicollinearity is not a serious issue in our analysis. The Shapiro-Wilk test was not able to reject the normality assumption of the model residuals at the 5% significance level. The violation of the homoscedasticity assumption was indicated neither by the Breusch-Pagan test nor by the White test [28].

Table I shows the resulting parameter estimates. We find strong support for our hypothesis that higher levels of conformance quality in OSS development projects are associated with lower failure costs. The results further indicate that, as expected, functional size has a significant adverse effect on failure costs. Since the regression coefficients in the log-linear model denote elasticities, a 1% increase in conformance quality is associated with about a 0.83% decrease in failure costs, whereas a 1% increase in functional size is associated with about a 0.80% increase in failure costs. The coefficient for project age is significant at a type I error level of 5%: An increased age tends to be associated with higher failure costs. No significant association between team size and failure costs could be observed.

IV. DISCUSSION

The results support our initial hypothesis that higher conformance quality is linked with lower failure costs. We thus revalidate the findings from closed source projects and production economics research [12, 13, 19, 20] and extend them to OSS projects.

As argued in Section II.B and evidenced in prior studies, functional size has a strong influence on project performance [21, 22]. Our study reveals that functional size has a significant impact on failure costs: As software size grows, failure costs increase. This can be explained by the fact that the functional size of a software product is related to its complexity. When a software grows in size, it tends to become more complex and hence more difficult to understand for a developer [21]. The difficulties in understanding software source code are rooted in the limited cognitive abilities of humans. Any human, regardless of his or her experience and knowledge, can only process and interpret a certain amount of information at once [21]. This implies that it takes a developer more effort to locate and understand the root of a software fault if the functional size is larger. Moreover, as the functional size increases the fault density of a software product often increases as well [12, 13]. As a consequence, there is a super-linear increase in the number of faults. Higher functional complexity, resulting in more faults that are more difficult to deal with, thus leads to higher failure costs.

Our study further indicates that team size is not significantly associated with failure costs. This finding is in contrast to prior studies reported in the software engineering literature [21, 22]. There, especially in the context of closed source commercial software development projects, team size was identified to have a significant influence on project performance during development and maintenance [21, 22]. The results of our study suggest something different for OSS development projects. It seems that team size does not have any negative effect on project performance. This may be explained by several facts: First, all team members are virtually distributed; hence there may be a higher awareness of potential problems in the collaboration process, and techniques for preventing them. Second, SourceForge.net offers a broad range of services supporting collaborative software development. Third, OSS teams are often composed of very talented developers far above average, which helps compensate negative performance effects [6].

The positive influence of age on failure costs seems unexpected if age is considered a proxy for virtual organizational learning [7]. One might assume that due to learning effects for longer-running projects higher age is associated with lower failure costs. In our study, this does not seem to be the case. A plausible explanation for our finding can be given when taking into account that in OSS projects developers tend to join and leave frequently [6]. The extent of learning on the part of the team members is therefore limited. Even worse, prior research has shown that in long-running projects, the software code gets increasingly complex and more difficult to maintain. This effect is sometimes referred to as software aging [29]. (However, note that the term “software aging” is not used consistently across research communities [30].) In an unstable team environment, the effort for understanding and correcting faults thus increases with project age, and so do failure costs.

V. CONCLUSIONS

In this paper, we have analyzed whether or not conformance quality and failure costs are associated in open source software projects. Our empirical results indicate that lower failure costs are linked with higher conformance quality and with smaller functional size of the software product. This is consistent with prior research and shows that the principles of production economics also hold for the growing field of OSS development. Increased project age tends to be linked to higher failure costs, whereas team size does not show any significant association with failure costs.

Of course, our study may have some limitations. We tried to ensure validity by relying on measures used in prior studies. However, better proxies than the ones chosen might exist. Another limitation could be the rather small sample size, although a set of around 30 observations is common in the field of production economics.

Further research should especially be devoted to the concept of project age/organizational learning. Being used as a control variable, this concept showed an unexpected (albeit weak) association with failure costs. In production economics, organizational learning has been proven to exert a strong positive impact on project performance. It would therefore be interesting to investigate these potential peculiarities of open source software development projects.

REFERENCES

[[1] J. A. Whittaker and J. M. Voas, "50 years of software: Key principles for quality," *IT Professional*, vol. 4, no. 6, pp. 28-35, 2002.

[2] P. Middleton and J. Sutton, *Lean Software Strategies: Proven Techniques for Managers and Developers*, New York: Productivity Press, 2005.

[3] RTI, "The economic impacts of inadequte infrastructure for software testing," National Institute of Standards and Technology, Gaithersburg, Planning Report 02-3, 2002.

[4] B. Boehm and V. R. Basili, "Software defect reduction top 10 list," *IEEE Computer.*, vol. 34, no. 1, pp. 135-137, 2001.

[5] G. von Krogh and E. von Hippl, "The promise of research on open source software," *Management Science*, vol. 52, no. 7, pp. 975-983, 2006.

[6] J. Bitzer and P. J. Schroder, *The Economics of Open Source Software Development*, Wakefield: Emerald, 2006.

[7] Y. A. Au, D. Carpenter, X. Chen, and J. G. Clark, "Virtual organizational-learning in open source software development projects," *Information & Management*, vol. 46, no. 1, pp. 9-15, 2009.

[8] H. G. Crokett, "Quality, but just enough," *Factory Management & Maintenance*, vol. 93, no. 6, pp. 245-246, 1935.

[9] L. M. Karg and A. Beckhaus, "Modelling software quality costs by adapting established methodologies of mature industries," in *Proc. 2007 IEEE IEEM*, pp. 267-271, 2007.

[10] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed., New York: McGraw-Hill Professional, 2001.

[11] D. Galin, *Software Quality Assurance: From Theory to Implementation*, Harlow: Pearson Education Limited, 2004.

[12] S. A. Slaughter, D. E. Harter, and M. S. Krishnan, "Evaluating the cost of software quality," *Communications of the ACM*, vol. 41, no. 8, pp. 67-73, 1998.

[13] N. Ramasubbu and R. K. Balan, "Globally distributed software development project performance: an empirical analysis," in *Proc. ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 125-134, 2007.

[14] A. R. T. Williams, A. Wiele, and B. G. Dale, "Quality costing: A management review," *International Journal of Management Reviews*, vol. 1, no. 4, pp. 441-460, 1999

[15] W.-H. Tsai, "Quality cost measurement under activity-based costing," *International Journal of Quality & Reliability Management*, vol. 15, no. 7, pp. 719-752, 1998.

[16] M. Grottke and C. Graf, "Modeling and predicting software failure costs," in *Proc. 33rd Annual IEEE Computer Software and Applications Conference*, pp. 180-189, 2009.

[17] D. G. Messerschmitt and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry*, Cambridge: MIT Press, 2003.

[18] T. L. Albright and H. Roth, "The measurement of quality costs: an alternative paradigm," *Accounting Horizons*, vol. 6, no. 2, pp. 15-27, 1992.

[19] L. P. Carr and L. A. Ponoemon, "The behavior of quality costs: classifying the confusion," *Journal of Cost Management Practices*, vol. 8, no. 2, pp. 26-34, 1994.

[20] V. K. Omachonu, S. Suthummanon, and N. G. Einspruch, "The relationship between quality and quality cost for a manufacturing company," *International Journal of Quality & Reliability Management*, vol. 21, no. 3, pp. 277-290, 2004.

[21] 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, no. 4, pp. 433-450, 1998.

[22] N. Ramasubbu, S. Mithas, M. S. Krishnan, and C. F. Kemerer, "Work dispersion, process-based learning, and offshore software development performance," *MIS Quarterly*, vol. 32, no. 2, pp. 437-458, 2008.

[23] J. Wu, K. Y. Goh, and Q. Tang, "Investigating success of open source software projects: A social network perspective," in *Proc.28th International Conference on Information Systems*, pp. 1-16, 2007.

[24] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?," in *Proc. 4th International Workshop on Mining Software Repositories*, p. 1, 2007.

[25] QSM. "Function point languages table," Version 3.0, 2005. Available: http://www.qsm.com/FPGearing.html

[26] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2009.

[27] J. B. Ramsey, "Tests for specification errors in classical linear least squares regression analysis," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 31, no. 2, pp. 350-371, 1969.

[28] W. H. Greene, *Econometric Analysis*, 6th ed., Upper Saddle River: Pearson Prentice Hall, 2008.

[29] D. L. Parnas, "Software aging," in *Proc. 16th International Conference on Software Engineering*, pp. 279-287, 1994.

[30] M. Grottke and K. S. Trivedi, "Software faults, software aging and software rejuvenation," *Journal of the Reliability Engineering Association of Japan*, vol. 27, no. 7, pp. 425-438, 2005.