The Effects of Development Team Skill on Software Product Quality

Justin M. Beaver
NASA, Kennedy Space Center
Justin.M.Beaver@nasa.gov

Guy A. Schiavone
Institute for Simulation and Training
Guy@ist.ucf.edu

Abstract

This paper provides an analysis of the effect of the skill/experience of the software development team on the quality of the final software product. A method for the assessment of software development team skill and experience is proposed, and was derived from a workforce management tool currently in use by the National Aeronautics and Space Administration. Using data from 26 small-scale software development projects, the team skill measures are correlated to 5 software product quality metrics from the ISO/IEC 9126 Software Engineering Product Quality standard. In the analysis of the results, development team skill is found to be a significant factor in the adequacy of the design and implementation. In addition, the results imply that inexperienced software developers are tasked with responsibilities ill-suited to their skill level, and thus have a significant adverse effect on the quality of the software product.

Keywords: software quality, development skill, software metrics

Introduction

Upon assignment of a new software development effort, a project manager begins to build the project's infrastructure in order to maximize the chances for project success. Does this project manager aggressively pursue the latest development process, and demand its implementation? Does this project manager define thresholds for measures of cyclomatic complexity [12] or lines of code in order to set boundaries on the complexity of the design solution? No. Typically, among the first actions a manager of a software development project will make will be to secure the services highly skilled and experienced team members. To the software project manager, this is strongest approach to project success.

Despite the intuitive relationship between development team skill and software product quality, the body of software engineering research is surprisingly sparse in its coverage of the topic. While very few software engineering models have attempted to capture development team skill as a driving cause of the success of software projects (e.g., [1]), the primary focus of most software quality models (e.g., [2][3][4]) is the analysis of design complexity measures as indicators of software quality. In practice, the software development industry has focused on well-defined and followed software processes as the key to software quality and project success [7] [8]. While it is difficult to argue that complexity and process are factors in the quality of a developed software product, there is a general failure to address what every software project manager knows all too well: the skill/experience of the software development team is a driving factor in software product quality.

James Bach has authored articles [5] [6] insisting that the quality of the people is the primary driver for software quality, and that too much industry focus has been on the process. He argues [5] that personal performance is "guided by higher level process models embedded within experience, education, and insight." Certainly from an intuitive and experiential perspective, the education and abilities of an individual play an important part in the ultimate quality of their developed software. However, there is little empirical evidence to support this logical assumption.

This research attempts to quantify the relationship between the skill of the software development team, and several measures of software product quality. It begins by proposing a framework for measuring the collective skill of a development team. Also, a description of several software product quality measures is provided including an overview of how those measures were obtained. An analysis of the degree of correlation between the development team skill measures and the software product quality measures is presented, and some inferences are made based on the underlying software engineering data.

Measurement of Development Team Skill

The assessment of the capability of the software development team requires insight into the team's collective education and experience. The assumption is that a more educated and experienced team will produce higher quality in their software product and artifacts. The construction portion of the software life cycle is comprised of three phases, each of which requires different types of skills/experience to accomplish successfully. The requirements phase is focused on organizational and interpersonal skills: interacting with customers, eliciting needs from those customers, and organizing the needs into a set of requirements. The design phase requires the ability to create a software solution that meets the identified requirements, and skills in evaluating the set of approaches and tools to implement that solution. Implementation demands a working knowledge and skill in the specific development tools, including languages, operating systems, libraries, and development environments. While integration/test is a vital part of the development life cycle, the focus of this study is on skill and experience in the "building" phases of development.

Because of the wide range of skills required at each life cycle phase, the approach to measurement of development team skill is to evaluate skill and experience of the team in terms of those life cycle phases. For example, a development team with vast design skill but little requirements development skill may produce a fantastic product, but may not necessarily meet the needs of the customer.

In this research, the collective skill of the development team is based on the skill ratings of individual team members. The individual skill ratings will be consolidated into a set of measures that reflects the distribution of skills for the entire team across the three life cycle phases. The National Aeronautics and Space Administration's (NASA) Competency Management System (CMS) [9] has been adopted as the approach to individual skill assessment. The CMS is a measurement framework, applied enterprise-wide, to quantify the knowledge, skills, and abilities of the NASA workforce, and to intelligently manage the allocation of that workforce those projects where appropriate needs exist. For each technical discipline (e.g., software engineering, chemistry, etc.), the CMS

establishes an ordinal, 4-tier scale for assessing an individual's knowledge, skill, and experience. For each of the four tiers associated with a discipline, a set of criteria has been developed which an employee must meet in order to be assessed at that level. In this research, the CMS approach has been adapted to apply in terms of the software engineering life cycle. A four-tier ordinal scale is used to rate an individual's skill/experience within each of the three building phases of the life cycle: requirements, design, and implementation. Associated with each tier, and for each phase, is a set of criteria which, if met, would allow an individual to be assessed the appropriate numerical rating for that phase.

Table 1 Collective Development Team Skill Measures

Phase	Skill Level	Measure Description
Requirements	Level 1	Proportion of the development team involved in requirements development that was assessed at Skill Level 1.
	Level 2	Proportion of the development team involved in requirements development that was assessed at Skill Level 2.
	Level 3	Proportion of the development team involved in requirements development that was assessed at Skill Level 3.
	Level 4	Proportion of the development team involved in requirements development that was assessed at Skill Level 4.
Design	Level 1	Proportion of the development team involved in software design that was assessed at Skill Level 1.
	Level 2	Proportion of the development team involved in software design that was assessed at Skill Level 2.
	Level 3	Proportion of the development team involved in software design that was assessed at Skill Level 3.
	Level 4	Proportion of the development team involved in software design that was assessed at Skill Level 4.
Implementation	Level 1	Proportion of the development team involved in software construction that was assessed at Skill Level 1.
	Level 2	Proportion of the development team involved in software construction that was assessed at Skill Level 2.
	Level 3	Proportion of the development team involved in software construction that was assessed at Skill Level 3.
	Level 4	Proportion of the development team involved in software construction that was assessed at Skill Level 4.

Although NASA's CMS provides the basis for an approach to individual skill assessment, it is necessary to broaden the individual ratings in order to capture the combined skill and experience of an entire software development team. Assessing skill for a development team involves consolidating the individual skill levels into a set of measurements that would accurately reflect the team's skill distribution in each of the life cycle phases, as first proposed in [11]. Table 1 lists the 12 metrics selected to represent a development team's skill and experience. For each phase and skill level, the proportion of the development team members that were assessed at that phase/skill level is the metric used. Normalizing the distribution of team skill by team size allows for the comparison of projects regardless of the number of team members. In addition, incorporating measures that independently address each life cycle phase allows for different team sizes and skill mixes in each phase to be accurately represented. It should be noted that no attempt is made in this research to analyze the composition of the development team in terms of personality, or in the light of team dynamics. While this perspective on team effectiveness is intriguing and seems a solid candidate for future research, the proposed skill assessment framework simply does not account for it.

Measurement of Software Quality

Software product quality in this research is captured using the ISO/IEC 9126 [10] as a guide for the expectations of quality within the delivered software product. The ISO/IEC 9126 is an international standard for software product quality that represents the quality of a delivered software product in terms of six major characteristics: Functionality, Efficiency, Reliability, Usability, Maintainability, and Portability.

Table 2 Software Product Quality Metrics

ISO/IEC 9126 Metric	Metric Purpose	Metric Description
Design Adequacy (DA)	How adequate are the checked design modules?	The ratio of the number of design modules evaluated to function adequately to the number of design modules.
Implementation Adequacy (IA)	How adequate are the checked source code units?	The ratio of the number of source code units evaluated to function adequately to the number of source code units.
Functional Implementation Completeness (FICMP)	How complete is the functional implementation?	One minus the ratio of the number of functions detected as missing during evaluation to the number of functions described in the requirements.
Functional Implementation Coverage (FICOV)	How correct is the functional implementation?	One minus the ratio of the number of functions detected as missing or incorrectly implemented during evaluation to the number of functions described in the requirements.
Functional Specification Volatility (FSV)	How volatile is the functional specification after baseline?	The ratio of the number of requirements changed after baseline to the number of requirements.

In the standard, each of the six quality characteristics is further partitioned into sub-characteristics and associated indicator metrics that allow for consistent measurement and assessment of quality. This research has focused on capturing and modeling product quality in terms of the Suitability of the product. Suitability is an ISO/IEC 9126 sub-characteristic of Functionality that quantifies the adequacy of the software product in terms of both its coverage of user needs and its implementation correctness. Table 2 lists the metrics used to capture software product quality for the Suitability portion of the ISO/IEC 9126 standard. These measures attempt to capture the level of quality of the software from the perspective of functional operation of the system. The only discrepancy between the metrics presented here and the ISO/IEC 9126 is that the measure of Functional Adequacy has been expanded to address adequacy of both design modules and source code units.

Experimental Approach

The approach to establishing the relationship between development team skill and measures of software quality was to record skill and quality measures from actual software development projects and analyze the results. A correlation coefficient was calculated to quantify the relationship between each of the skill variables, and each of the software product quality metrics. The equation used for the correlation coefficient is shown in Equation 1.

Equation 1 Correlation Coefficient Calculation

$$Correl(X,Y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2 \sum (y-\bar{y})^2}}$$

The sets of data used for this analysis are from 26 software development projects. The projects were small in scale, and were generally completed within a 3-4 month time frame. Project teams varied in size from 1 to 4 developers and included both graduate students and software engineering professionals. The projects selected had sequentially addressed each phase of the development life cycle in a classic "waterfall" fashion. Of the original 35 projects selected to participate in this research, 26 were actually used because of their willingness to track life cycle measures completely and correctly.

Using the measurement frameworks outlined above, it is appropriate to hypothesize the results of the analysis in terms of correlating development team skill to software product quality. Table 3 captures the expected relationship between the Development Team skill levels and the software product quality metrics. The expectation of a direct relationship between the increasing skill levels and the increasing correlation is annotated in Table 3 as "Incr.". For example, as the design skill increases, it would be expected that the correlation between that skill value and the Design Adequacy metric would increase. This indicates a positive relationship and the influence of the higher level of skill and experience on the design of the system. The "Decr." annotation captures the expectation that an inverse relationship, or decrease in correlation, exists for the increasing skill levels. Requirements Volatility provides a good example of this case. As requirements skill increases, it is intuitively expected that the volatility of the requirements will decrease.

Table 3 Expectations for Correlating Skill to Software Quality

Phase Skill	ISO/IEC 9126 Software Product Quality Metric				
	DA	IA	FICMP	FICOV	FSV
Requirements Skill	Incr.	Incr.	Incr.	Incr.	Decr.
Design Skill	Incr.	Incr.	Incr.	Incr.	Decr.
Implement Skill	N/A	Incr.	Incr.	Incr.	Decr.

The sequential nature of the software life cycle implies a path of dependency from one phase to another. That is, the quality in products of the requirements phase affects the quality of products in the design phase since design products are derived from requirements products. Similarly, we would expect that a high level of skill in the requirements phase would affect not only the requirements phase products, but also design phase products. This relationship is also captured in Table 3. Conversely, the sequential nature of the life cycle prevents downstream activities from directly affecting upstream activities. Thus, Implementation Skill was not a factor in the development of the design.

Results

Table 4 displays the correlation coefficients that relate the measures of development team skill to the selected measures of software quality. An examination of the results reveals some expected relationships, and also some interesting deviations from the expected correlation values.

Table 4 Correlation Coefficients of Skill to Software Quality

Skill Level	ISO/IEC 9126 Software Product Quality Metric				
	DA	IA	FICMP	FICOV	FSV
Rqmnt 1	-0.415	-0.755	-0.022	-0.146	-0.062
Rqmnt 2	-0.419	-0.619	-0.042	0.038	0.190
Rqmnt 3	0.434	0.752	-0.013	-0.030	-0.094
Rqmnt 4	0.515	0.645	0.266	0.411	-0.212
Design 1	0.045	-0.314	0.246	0.166	0.052
Design 2	-0.513	-0.592	-0.302	-0.404	0.158
Design 3	0.433	0.693	0.059	0.219	-0.152
Design 4	0.413	0.358	0.288	0.375	-0.235
Impltn 1	N/A	-0.377	0.258	0.129	0.085
Impltn 2		-0.758	-0.406	-0.296	0.168
Impltn 3		0.367	-0.017	-0.094	-0.058
Impltn 4		0.597	0.307	0.375	-0.252

Effects of Skill on Adequacy

As expected, the increased presence of more highly skilled software developers on the development team was positively correlated with the quality measured in the final software product. This was most evident in the correlation results of the various skill levels to the measures of Design and Implementation Adequacy. There is a sharp contrast between the correlation of the highly skilled (Levels 3 and 4) developers to adequacy, and the correla-

tion of the marginally skilled (Levels 1 and 2) developers to Adequacy. The increasing presence of the less skilled in each phase of the development life cycle has a significant negative correlation to the adequacy of the design and implementation. The practical inference made from these relationships is that increasing numbers of less skilled developers *decrease* the quality of the software product in terms of its adequacy. Conversely, the increasing presence of more skilled developers has a significant positive correlation to the adequacy of the software product, implying that increasing numbers *improve* the quality of the final product. These correlations provide empirical evidence for the intuitive relationship between development team skill/experience and software quality.

Effects of Skill on Specification Volatility

The correlation coefficients associating development team skill to Functional Specification Volatility were as expected in terms of their inverse relationship. That is, as skill levels increased, there was a general decrease in the volatility of the functional specification. Despite the expected behavior, the magnitude of the correlation coefficients is surprisingly low. Intuitively, one would expect volatility to decrease dramatically as the collective skill of the requirements team increased. The low correlation coefficients indicate that requirements skill has very little impact on the volatility of the software specification. Further research is required, possibly exploring the effects of software development process or product complexity, to identify a more significant factor that affects Functional Specification Volatility.

Adverse Impact of Skill Level 2

The most prominent unexpected result is the divergence of Skill Level 2 from the expected increasing and decreasing correlation patterns. For each quality attribute, and particularly with respect to design and implementation skill, the increased presence of personnel with Skill Level 2 has an adverse impact on that quality attribute. For example, Design Skill Level 2 is expected to be more positively correlated than Design Level 1 to the adequacy, completeness, and coverage quality metrics. However, for each of those metrics, Design Level 2 is discovered to be significantly more negatively correlated than Design Level 1. In three of the four cases the difference is on the order of 0.5. Similar results exist for Implementation Skill Level 2.

The unexpected adverse impact of Skill Level 2 may be the result of a flaw in the assignment of skill levels to personnel. Projects participating in this research were required to have their developers complete a questionnaire from which each developer was assigned a skill level based on their responses. It is possible that the set of criteria that assesses personnel at Skill Level 2 is flawed in that developers are incorrectly rated. The existence of deviations from the expected for Skill Level 2 in all life cycle phases is further evidence that the set of criteria may need refinement. However, a flaw in assignment criteria should also cause unexpected results in the correlation values for the other skill levels. Since this was not the case, we propose a different reason for the results.

Another explanation is that the deviation of Skill Level 2 is simply an indication that the presence of less skilled developers has a negative influence on a development effort. This premise must be explored further, however, as the same dramatic deviation was not noticed for the correlation values associated with Skill Level 1. In a software development effort, the minimally skilled (e.g., Skill Level 1) are easily recognized, and often assigned more trivial aspects of the design and implementation in order to improve their individual skill/experience. However, those with a marginal skill set or with a small amount of experience (e.g., Skill Level 2) are not as appropriately allocated to easier tasks. A developer with basic competence in design and implementation is often assigned an equal share of the development responsibilities with his/her more highly skilled colleagues. It is proposed that those developers that have a small amount of software development ability (Skill Level 2) are given software development tasks that are inappropriately matched to their skill/experience level. For that reason, the increased presence of Skill Level 2 has an adverse negative impact on the quality of the software development product.

Effects of Requirements Skill

The effect of Requirements Skill on software product quality was most pronounced in its effect on Design and Implantation Adequacy. The presence of higher requirement skill levels had a strong correlation to the adequacy of the design and implementation, while the presence of lower requirement skill levels had an opposite and negative effect on adequacy. Increasing numbers of engineers that are unskilled in requirements development appear to impair a project's ability to provide adequate design modules and source code units. This result supports the notion that the quality of the requirements has a dramatic effect on its downstream products in the design and implementation phases of development.

In analyzing the effect of requirements skill on the Functional Implementation Coverage and Completeness, it is interesting to note the lack of significance of skill levels except for Skill Level 4. It appears that only the involvement of the most highly skilled requirements engineers has a significant effect on the correctness and completeness of the final software product. This may be an indication of leadership on the requirements development team. Leadership in developing requirements was a criterion for assessing individuals at Requirements Skill Level 4. It is reasonable to infer that experienced leadership in requirements development effort is the driving factor in the effect of requirements skill on the Functional Implementation Completeness and Coverage.

Effects of Design Skill

The Design Skill measures correlated most significantly with the adequacy of the design and implementation. This aligns with the intuitive assumption that those with a higher level of design skill and experience will produce a more adequate design, and therefore a more adequate implementation. Similarly, those with less design skill and experience will produce a less adequate design and implementation.

In addition, Design Skill Level 4 was moderately positively correlated with the functional implementation completeness and coverage quality variables. Similar to the inferences made in the analysis of the effects of requirements skill, it is postulated that design leadership is the skill element that distinguishes Design Level 4 as a significant factor for these quality attributes.

Effects of Implementation Skill

The measures of Implementation Skill correlated most significantly to implementation adequacy. As with the design and requirements

skill, increasing numbers of less skilled developers were negatively correlated to Design and Implementation Adequacy, and increasing numbers of more skilled developers were positively correlated to those quality attributes. Implementation Skill Level 4 had a moderate correlation to the functional implementation completeness and coverage metrics.

Threats to Validity

All of the software engineering data used in this report is from projects that are relatively small, object-oriented software efforts. While the results provide evidence that development team skill is correlated to software product quality, it is premature to infer that the results are applicable outside of the scope of the underlying software engineering data. Future research is required to validate these results for both large-scale projects, and projects that do not use the object-oriented paradigm.

Conclusion

This report provided empirical evidence that development team skill has a significant effect on the quality of a software product. A measurement framework for assessing software development skill and experience was used to quantify the capabilities of the software development team. These measures were correlated to measures of software product quality taken from the ISO/IEC 9126 standard. As expected, higher proportions of skilled and experienced staff in the requirements, design, and implementation life cycle phases increased the adequacy of the design modules and source code units, and had a similar albeit more muted effect on the functional implementation completeness and correctness, and on the volatility of the software specification.

In general, higher proportions of skilled engineers had the most dramatic effect in terms of adequacy of the design and implementation. Conversely, higher proportions of less skilled engineers negatively affected the end product quality of the software. The adverse impact of Skill Level 2 on the software quality metrics implies that marginally skilled developers are assigned work that is inappropriate for their skill levels. For the relationship between development team skill and Functional Implementation Coverage and Completeness, the increased presence of experienced leadership (Skill Level 4) in each of the life cycle phases had the most significant impact.

The proposed framework for measuring development team skill and experience provided an adequate representation of the distribution of individual skills on a given development team. It is recommended that software project managers assess their software developers using the proposed framework in order to better understand the strengths and weaknesses of their staff. Such an understanding could prevent the assignment of tasks that are not well suited to less skilled and experiences engineers, and thus prevent the negative impact on product quality.

Future work for this research is planned. The method of skill assessment will be revisited in order to verify that the application of the skill criteria correctly assigns skill levels to project personnel. In addition, data collection for a more diverse set of projects in terms of size and complexity is expected. This research also will be expanded to provide a basis for staffing software engineering projects. The focus will be the determination of optimum skill/experience combinations, taking into consideration personnel costs and resource availability, in order to maximize the quality on all software projects.

References

[1] B.W. Boehm, J.R. Brown, and M. Lipow. "Quantitative Evaluation of Software Quality." In *Proceedings of the 2nd International Conference on Software Engineering*, pp. 592-605, Los Alamitos, CA, 1976. IEEE Computer Society Press.

[2] V.R. Basili, L.C. Briand, and W.L. Melo. "A Validation of Object-Oriented Design Metrics as Quality Indicators." *IEEE Transactions on Software Engineering*, **22**(10):751-760, October 1996.

[3] T.M. Khoshgoftaar, J.C. Munson, B.B. Bhattacharya, and G.D. Richardson. "Predictive Modeling Techniques of Software Quality from Software Measures." *IEEE Transactions on Software Engineering*, **18**(5):979-987, November 1992.

[4] R. Subramanyam and M.S. Krishnan. "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects." *IEEE Transactions on Software Engineering*, **29**(4):297-310, April 2003.

[5] J. Bach. "Enough About Process: What We Need Are Heroes." *Computer*, **12**(2):96-98, March 1995.

[6] J. Bach. "What Software Reality Is Really About." *Computer*, **32**(12):148-149, December 1999.

[7] M. Diaz and J. Sligo. "How software process improvement helped Motorola." *IEEE Software*, **14**(5):75-81, Sept.-Oct. 1997.

[8] T.J. Haley. "Software process improvement at Raytheon." IEEE Software, 13(6):33-41, November 1996.

[9] NASA. "Competency Management System [online]." In http://ohr.gsfc.nasa.gov/cms/home.htm, Goddard Spaceflight Center, MD, April 2005. National Aeronautics and Space Administration.

[10] ISO/IEC 9126. ISO/IEC 9126:2001. *Software engineering - Product Quality*. International Organization for Standardization, Geneva, Switzerland, 2001.

[11] J.M. Beaver, G.A. Schiavone, and J.S. Berrios. "Predicting Software Suitability Using a Bayesian Belief Network." In *Proceedings of the 4th International Conference on Machine Learning and Applications*, pp. 82-88, Los Angeles, CA, 2005. IEEE Computer Society Press.

[12] T. McCabe. "A Complexity Measure." *IEEE Transactions on Software Engineering*, SE-2(4):308-320, December, 1976.