Standards are Necessary but not Sufficient for Excellent Software

David N. Card

card@computer.org

Abstract

Software is becoming increasingly pervasive in business systems and everyday life. Software is the glue that holds together complex systems. It is impossible to talk about the quality of a software dependent system without discussing software quality. A software dependent system is a system that can't perform its intended function if the software doesn't work correctly. However, since software is invisible, systems developers often do not give it sufficient attention. Nevertheless, inadequate quality can put businesses and lives at risk.

This paper provides a systematic overview of software quality in the context of a system. This includes static, dynamic, and customer/user perspectives. Quality standards have been widely adopted in recent years. However, these standards only address static aspects of quality, e.g., conformance to requirements. Moreover, the author's experience shows that these standards are often incompletely implemented. This is facilitated by a lack of appropriate feedback on the quality status of software products. As described here, objective measurements and independent in-process audits can be part of the solution.

Biography

David N. Card is an independent consultant working in the areas of software quality assurance for aerospace, automotive, and marine systems. He is also a research associate of the Experimental Software Engineering Group of the Universidade Federal do Rio de Janeiro, Brazil. Previous employers include Computer Sciences Corporation, Det Norske Veritas, and Lockheed Martin. Mr. Card is the author of many articles and two books on the topics of software quality, reliability, estimation, and performance management. He has worked with scores of software engineering organizations seeking to improve their performance. He does not promote any single methodology. He has also participated in many project reviews and accident investigations.

1 Introduction

The term “software quality” has been redefined so many times that its meaning has become unclear. In this paper we introduce the term “software excellence” to encompass most interpretations of software quality. Excellent software meets requirements and is defect-free, safe, reliable and secure from cyberattack. Figure 1 identifies these properties. Note that they are not orthogonal dimensions. They are overlapping and dependent. For example, research (Woody, 2014) shows that 50% of security exploits are linked to defects introduced during development and maintenance. Consequently, establishing an engineering process that minimizes defects is critical to achieving software excellence. The practices of an effective engineering process are defined in process models and standards (e.g., International Standards Organization, 2017)

2 How do Standards and Models fail?

The Capability Maturity Model – Integration (CMMI) (Chrissis, et al.) is popular in the aerospace and automotive industries. The CMMI is used to assign a maturity rating or score to an organization’s software engineering process. However, organizations that are assessed as high maturity often do not operate with high performance, especially with regard to quality. There have been several significant examples in recent years, although these are not discussed publicly.

How does this happen? Software and systems development are human intensive activities, typically conducted in complex organizations. Established past performance may predict future performance, but good behaviors (maturity) may not be maintained in situations of stress. The CMMI assumes that the QA role will maintain quality performance, but project management usually has greater power while focusing on cost and schedule. Integrated Software Dependent Systems (ISDS) (Det Norske Veritas, 2012) is similar to the CMMI but is intended to be used to monitor ongoing software development processes. It does not assume that because an organization can correctly answer a questionnaire or has demonstrated good performance in the past that it will deliver good performance on the current project.

Another problem is that process models may be incomplete and/or poorly implemented. This may be due to a misunderstanding of process requirements. For example, the concept of “peer reviews” is well define in software engineering literature, but I have seen examples of implementations ranging from peer reviews that find no defects to peer reviews of the same artifact repeated multiple times.

Software development is a human activity subject to human weaknesses such as misunderstanding and overconfidence, Experienced developers often believe that their skill enables them to take shortcuts without suffering consequences. ISDS addresses this by requiring in-process audits.

A current issue is the myth that Agile eliminates the need for processes. The Agile manifesto (Beck, et al., 2001) specifically de-emphasizes processes (Individuals and interactions over processes and tools), but that does not mean that processes are unnecessary. In fact, rigorous processes (especially Configuration Management) are essential to an effective Agile project, where different parts of the product are being developed and tested at different times, with the expectation that they will all integrate easily at the end. The most effective Agile organization I have encountered developed software for cable TV set top boxes. They followed a disciplined process that incorporated Agile practices such as Scrums and Sprints. The Quality Assurance organization provided intensive coaching on the methods to be employed and Project Management enforced the process regardless of cost and schedule challenges.

3 Nature of Quality

Quality may be viewed from many perspectives. Figure 1 illustrates some of them. This discussion focuses on software, but applies to software dependent systems, in general. A similar approach (de Rocha and Travassos, 2017) also includes the factor of service quality, which may be important for some products. Figure 1 shows that achieving software excellence requires taking static, dynamic, and operational perspectives to ensure high performance.

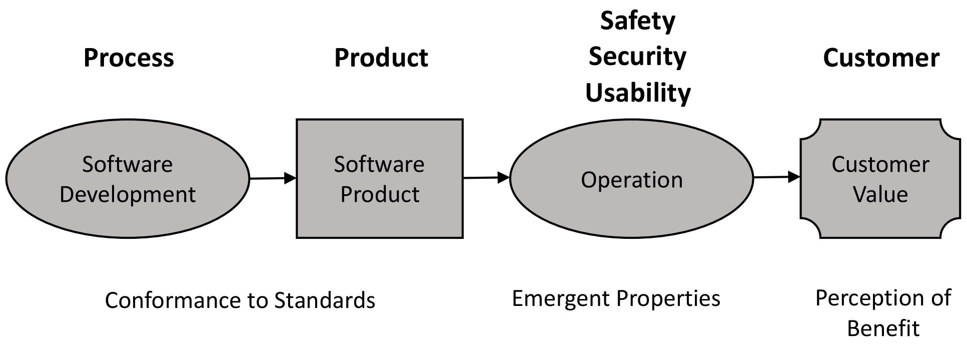

Figure 1 – Model of Contributors to Software Excellence

The software engineering process may be more or less capable of producing a good product. Process quality may be defined in terms of process standards or process reference models based on “best” (or at least “good”) practices. Of course, in order to provide benefit, these practices must be implemented by the developers.

The software product, itself, may be evaluated in terms of conformance to appropriate product standards. The process perspective has proven to be more important for software quality than for typical manufacturing industries, because software product standards have proven harder to agree upon, and logically, different standards are needed for different products. Software product standards typically define characteristics that are associated with successful operational use and maintenance of the software. Software product standards commonly focus on user interface and networking issues. Data may also be considered a software product that can be standardized for specific applications such as maps

The operating software may be evaluated in terms of the quality of service provided, such as security, safety and usability. As emergent properties, these qualities are not amenable to standards. They must be evaluated in the overall system context. For example, the safety of a software component depends on how the user interacts with it as well as the mechanical equipment attached to it. Conformance to product standards (such as “no single point of failure”) cannot ensure safety, but can make it easier to achieve. While Safety is a special concern for offshore vessels, most safety analyses assume that software is 100% reliable, but reliability also depends on the context of use. Security has largely been ignored in this industry. For example, many suppliers deliver software with a hard-coded password. Nevertheless, quality (as freedom from defects and nonconformance to requirements), is a pre-requisite for both safety and security. Research shows that 50% of all security vulnerabilities are due to software defects (Woody, 2014). Clearly, software quality has an immense effect on the operator's and customer's perception of the product, even if it is not the only factor.

The customer's satisfaction is based on his experience with the product as compared to his expectations of it. Satisfaction is a state of mind, not a physical state. Managing customer expectations is as important in achieving customer satisfaction as managing operational quality. Establishing standards for customer satisfaction is very difficult.

As indicated in Figure 1, the software process and software product are the views of software quality that are most amenable to standards. How do standards fit into software quality assurance? Standards provide expectations of activities to be performed or outcomes to be achieved. Satisfaction of these expectations can be confirmed by direct observation (verification), i.e., in-process audits.

4 Recommended Solution

In addition to in-process audits, appropriate measurements should be to monitor whether or not real quality performance meets expectations. Simple data from peer reviews (defects, effort) tell us about the performance of design and code activities. The author observed two illustrative cases where the number of reported defects from peer reviews was low. Audit and observation of the peer review activities in these projects showed that in one case (Case1) the peer reviews were pro forma – peer reviews lasted five minutes and few defects were recorded. In the other case (Case 2), the team conducted two rounds of peer reviews – in the first round defects were found and recorded; the second peer review occurred after defects had been fixed. Both teams were trying to do a good job, but in Case 1 the objective of peer reviews was not achieved, while in Case 2, the cost of peer reviews was doubled and important defect data was lost. A program of regular in-process audits would have led to earlier detection and correction of these situations.

The message is that simple data helped to identify that something unusual was going on, but that it was necessary to directly observe the behaviors in order to diagnose and correct the problem. Thus, our focus on measurement and in-process audits to ensure effective implementation of processes. Requirements Traceability is another activity where there are many technical approaches, but where developers may lose sight of the objectives.

In order to maintain a focus on quality, defects should be monitored throughout development. The author has had success with a Weibull modelling approach for agile and highly iterative processes. The Weibull model is suited for processes where defects are continually being introduced and removed from the system, but at different rates (Card, 2002). Figure 2 shows an example of a Weibull analysis of data from implementation and testing of a control system for marine navigation based on a standard design. The Weibull curve represents the theoretical optimum (i.e., frontier) performance of this verification process. However, in any given testing interval there are likely to be inefficiencies, e.g., holidays, delays, poor planning – so the actual performance tends to fall below the optimal performance leading to a "frontier" behavior (Zu, et al., 2001). In this case regression on the actual data, e.g. (Dola, et al. 2014) will not give us the correct (frontier) Weibull model. Instead we *impute* the Weibull parameters using the criteria that the observed peak of the distribution should coincide with the peak of the theoretical model, and the slope of the observed data should match the slope of the theoretical model. (The Delta Delta line in Figure 2 compares the slopes.) As both curves pass through the origin we can identify Weibull parameters to estimate the tail of the frontier model. The result shown in Figure 2 indicates that additional testing and fixing is needed to reduce the defect level and increase reliability.

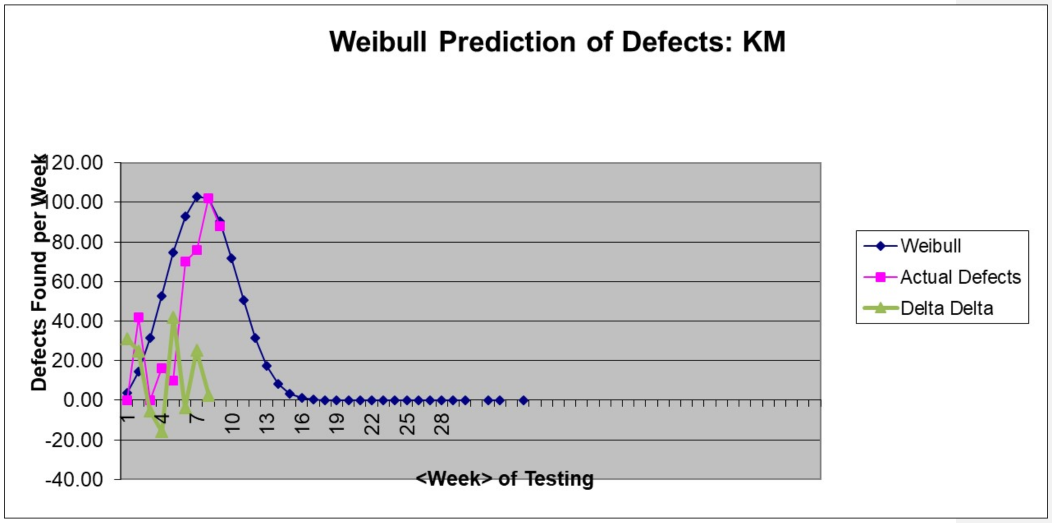

Figure 2 – Weibull Analysis of Defects

Sharma, Manini 7/17/2019 10:57 PM
Comment [1]: Chart axis titles are overlapping the chart impacting readability

5 Summary

Achieving software excellence in a complex system requires many different activities, not just implementation of standards and process models. Moreover, implementation of process models must be re-enforced through timely feedback. Agile processes require re-enforcement, too. Organizations that make major investments in methods and tools often fail to invest in feedback mechanisms to make sure that they are effectively implemented.

Process models and standards are necessary, but not sufficient, for excellent software. FAA standards require rigorous development and testing activities similar to those of the CMMI. However, failure to update a safety analysis after a late design change can result in disaster.

Similarly, rigorous development and testing of an offshore safety system that ignores how the operators can/will react to anomalous situations can result in failure to avoid disaster.

Nevertheless, good implementation of standards and process models creates an environment where safety, security, and operations studies can be successful. These processes ensure that artifacts (e.g., designs) necessary for FMECAs and usability studies are available.

References

D. Card and L.G. Chua, Ensuring Software Reliability, Safety, and Security, *Proceedings: RINA international Conference on Computers in Shipbuilding*, 2017

A.R. de Rocha and G.H. Travassos, QPS – Modelo de Referencia para Avaliacao de Produtos de Software, *Universidade Federal do Rio de Janeiro*, 2018

WOODY, C., *Predicting Software Assurance Using Quality and Reliability Measures*, Software Engineering Institute, 2014

Stamatis, D H.*, Failure Mode and Effects Analysis*, ASQ Press, 2003

Ogale P., M. Shin and S. Abeysinghe, Identifying Security Spots for Data Integrity, *Proceedings: IEE Computer Software and Applications Conference*, 2018

Det Norske Veritas (DNV), OS D-203, *Integrated Software Dependent Systems (ISDS)*, December 2012

CARD, D., A Software Integration and Process Model for Offshore Vessels, *Proceedings: Asia-Pacific Software Engineering Conference*, 2013

CHRISSIS, M., M. PAULK, and S. SHRUM, *Capability Maturity Model - Integrated*, Addison Wesley, 2003

Dola D.R., M.D. Jaybhaye, and S.D. Deshmukh, Estimation of System Reliability using Weibull Distribution, *International Proceedings of Economic Development and Research*, 2014

Zu T., M. Wen, and R. Kang, An optimal evaluating metho for uncertainty metric in reliability based on uncertain data envelopment analysis, *Microelectronics Reliability*, August 2017

Beck, Kent, et al., https://agilemanifesto.org/, 2001

International Standards Organization, ISO/IEC 12207, *Systems and Software Engineering: Software life Cycle Processes*, 2017

Card, D., Managing Software Quality with Defects*, IEEE Computer Software and Applications Conference*, 2002