ISSN 2229-5518

Valuing Software Quality Analysis

Meer Shizan Ali

Abstract— Software quality is the major challenge for all the Software Systems today. The requirements once necessary for Defense Systems are now equally challenging for everyday systems.Such complex projects usually involve a large and heterogeneous group of stakeholders with various quality priorities which are time varying and conflicting making both one size fits all quality metrics and software development driven by such metrics risky to use. This primary goal of this dissertation is to develop a Value based Software Quality Analysis Framework that integrates the development process aiming at achieving the appropriate quality levels for Software Systems and Stakeholder approach into quality attribute definitions, metrics and models.

Index Terms— Quality Assesment Frameworks,Software Quality, Software Quality Analysis, Software Quality Metrics, Value based Software Engineering, Value based Software Quality Analysis Framework, Value based Software Quality Achievement Process.

1 INTRODUCTION

THE early institutional focus on software quality was based on the requirements driven, contract-oriented waterfall model software development. But the major drawback of this approach is that Quality Assurance is based on initial contrat, if the contract specifies incomplete quality requirements we will get poor quality. .

Then came the service oriented customer satisfaction as the primary quality objective. Total Quality Management, Quality Function Deployment were based on expectations of customers who were generally interpreted to include users of software products. But the major difficulty with the customer satisfaction approach is that customers often lack a complete view of tradeoffs and interactions among their concerned software quality attributes, and they often neglect other quality attributes such as maintainability on which they are indirectly dependant on. Initiatives to address these problems in have focused on identifying a full set of success-critical stakeholders in a software system and pursuing the objective of negotiated stakeholder win-win relationships among software quality attributes. This has led to new organizational approaches such as Integrated Product Teams, expanded versions of QFD [Pardee 1996], and process approaches such as the WinWin Spiral Model [Boehm et. al. 1995a]. The future trends of software system development include an increased emphasis on stakeholders and end value, along with increasingly rapid change. However tha major challenges on software quality assessment implied by the future trends of software development are: The Universal one-size-fits-all software quality metrics are unachievable in most project situations, Stakeholder value dependencies on software quality attributes are often in conflict and require negotiated solutions, Stakeholders have often-emergent time-varying priorities for software quality attributes, Multi-criterion decision solutions are complicated by tradeoff relations among software quality attributes, and there is a need for better value estimating relationships for software quality attributes.

This paper will address the problem of software quality modeling and achievement process using value based approach examining the value based definitions of software quality attributes and applications of the models. It also outlines to define the software quality attributes from stakeholder's perspective and to measure the achievement of software quality attributes.

• *Meer Shizan Ali, Assistant Professor MIT, Indore*
E-mail: mshizan@gmail.com

2 RELATED WORK

The major challenges include the lack of a well-formalized and tangible definition of software quality and stakeholder / value-relevant software quality metrics and models for project decision-makers. Furthermore, how to use the metrics and models to drive the development processes on software quality achievement has not been sufficiently explored either.

2.1 Software Quality Attributes

The traditional definitions of each attribute of software quality as an independent sub-discipline of software quality is followed. *Reliability* is the ability of a system or component to perform its required functions under stated conditions for a specified period of time [IEEE 1990]. More definitions of Reliability and its factors are covered in [Rus et. al. 2003]. In [Leveson 1995], Safety is defined as *"freedom from accidents or losses"*. The IFIP WG 10.4 defines dependability as the property of a computer system such that reliance can justifiably be placed on the service it delivers [IFIP WG10.4]. Other related work on software quality attributes includes the TRW Characteristics of Software Quality study performed for the National Bureau of Standards [Boehm et. al. 1978] which provides a general framework for reasoning about software quality attributes; Gilb's work on quality attribute specification and management [Gilb 1976, Gilb 1988]; and the Fraunhofer-Maryland series of dependability attribute reports [Rus et. al. 2003].

2.2 Software Quality Metrics, Assessment frameworks

Traditional ways to measure Reliability falls into two classes: 1) Time-based definitions (failures per unit time, mean time to failure, mean time between failures, etc); 2) Domain-based

(failure/1000 pages printed, failures/1000 transactions, etc) definitions that are also dependent on the applications. Limitations with such metrics for software quality attributes are that not all failures have the same impcact and they are value-neutral and scenario-dependent.

The classical reliability models summarized in [Lyu 1996] include the follows: (1) Exponential Failure Time Class of Models (Jelinski-Moranda De-eutrophication Model, Nonhomogeneous Poisson Process (NHPP) Model, Schneidewind's Model, Musa's Basic Execution Time Model) (2) Weibull and Gamma Failure Time Class of Models (Weibull Model, S-Shaped Reliability Growth Model) (3) Infinite Failure Category Models (Duane's Model, Geometric Model) (4) Bayesian Models (Musa-Okumoto Logaritmic Poisson, Littlewood-Verrall Reliability Growth Model). Madeira and Koopman in [Madeira 2001] try to develop a benchmarking framework for dependability evaluation based on the IFIP WG 10.4 dependability exploration.Wilson et. al. in [Wilson 2002] seek to enable comparison of different computer systems in the dimensions of availability, data integrity, disaster recovery and security. Arlat in [Arlat 2001] describes a benchmarking framework specification for the availability attribute of software quality.

The two most compatible and complementary with our research objectives is that both of them argue about the importance of stakeholder involvement into the dependability assessment: Huynh et al. in [Huynh 2003] propose a Center of Mass (COM) model to represent their view of software quality/dependability as a multi-attribute and multi- stakeholder concept; and The Unified Dependability Model (UMD) proposed in [Basili 2004] aims to establish a common languagefor discussing a variety of software quality/dependability attributes and to make them measurable. However neither COM model nor UMD discusses how to use the metrics and model to drive the software quality achievement process.

2.3 Value-Based Software Engineering

The level of quality is multidimensional: for example, some applications depend on low latency but can tolerate low precision; in other applications precision is critical but latency is not. It follows that software that is acceptable in one situation may be deficient in another [Shaw 2002]. At the Economics-Driven Software Engineering Research Workshop [Sullivan et. al. 1999-2005] and elsewhere [Reifer 2002, Nejmeh 2002], researchers have developed general frameworks for making software engineering decisions about enhance the value of delivered software systems. [Butler 2002] develops a technique for selecting an appropriate suite of security technologies for a particular computer installation. Different installations must protect different resources; they have different budgets and different concerns about security threats.

In [Aurum et. al. 2005], Boehm presents an overview and agenda for Value- Based Software Engineering (VBSE). He discusses the seven key elements that provide candidate foundations for value-based software engineering with a case study. The four additional theories that it draws upon are utility theory (how important are the values?), decision theory (how do stakeholders' values determine decisions?), dependency theory (how do dependencies affect value realization?) and control theory (how to adapt to change and control value realization?). Other value-based view of software quality includes [Emam 2003], which investigates the Return On Investments (ROI) of software quality. It emphasizes the pre-release and post-release cost savings due to defect reduction in calculating the profit and ROI.

3 Value Based Software Quality Analysis Framework (VBSQA)

Despite the development of various software quality attribute modeling techniques and their related analyses, the proper definitions and modeling of software quality attributes from the perspectives of stakeholders' value propositions appear to be lacking. So we propose the stakeholder/value-based approach to leverage the software quality modeling and analysis. These also enable us to use the value-based software quality attribute definitions and models to drive a software development process to achieve stakeholder mutually satisfactory software quality requirements.

3.1 Stakeholder View of Software Quality

Complex projects involve a large and heterogeneous group of stakeholders with various quality perspectives and different needs. Ideally, one would like to have a single quality metric by which the development process could be driven. However, in practice, such a one- size-fits-all metric is unachievable. Different systems have different success-critical stakeholders, and these stakeholders depend on the system in different ways. Acritical first step in understanding the nature of software quality is to identify the success-critical stakeholder classes for a software system, and to characterize the relative strengths of their dependencies on various attributes of a given information system. This involves answering three main questions: What are the primary quality attributes of a software system that success- critical stakeholders depend on?, What classes of stakeholders exhibit different patterns of dependency on these attributes?, For each class of stakeholder, what is the relative strength of their dependency on each attribute?. Thus we aim to propose a software development process as a guide to help project stakeholders achieve their expected/desired levels of quality attributes using the value-based quality metrics, models and methods.

3.2 VBSQ Attribute Definitions

Our value-based definitions of safety, security, and privacy are as follows: A system provides *Safety* to the extent that it minimizes stakeholders' expected loss of value due to death, injury, illness, or damage to equipment, property, or the environment. A system provides *Security* to the extent that it minimizes stakeholders' expected loss of value from unauthorized access, use, disclosure, disruption, modification, or destruction of information assets, including financial losses and loss of value due to death, injury, illness, or damage to equipment, property, or the environment. A system provides *Privacy* to the extent that it minimizes stakehold-

ers' expected loss of value from authorized or unauthorized access, use, disclose, or modification of stakeholders' personal information, including financial losses and loss of reputation. A system provides *Reliability* to the extent that it maximizes the probability that the system will provide stakeholder-desired levels of service wrt a system's operational profile over a given period of time. A system provides *Availability* to the extent that it maximizes the fraction of time that the system will provide stakeholder-desired levels of service with respect to a system's operational profile. A system provides *Survivability* to the extent that it maximizes the total expected value obtained from achieving stakeholder-desired levels of service and from reduced levels of service when the desired levels of service are unachievable.

A system provides *Performance* to the extent that it maximizes the value of processed information achievable within the available resources being used to process the system's workload. A system provides *Accuracy* to the extent that it minimizes the difference between delivered computational results and the real world quantities that they represent. A system provides *Usability* to the extent that it maximizes the value of a user community's ability to benefit from a system's capabilities with respect to the system's operational profile. The oher attributes involved are: Evolvability, Interoperability, Correctness, Timeliness, Affordability, and Resuability.

3.3 Value Based Software Quality Model (VBSQM)

Different stakeholders depend on different quality attributes in different ways under different situations. The lack of good return-on-investment (ROI) models for software quality causes difficulties for decision-makers in determining the overall business case for software quality investments, in determining which software quality investments are most cost-effective, and in determining how much software quality investment is enough. We develop an integrated Value-Based Software Quality Model (VBSQM) for reasoning about software quality's ROI and performing combined risk analyses using the COCOMO II [Boehm et. al. 2000a], COQUALMO [Steece et. al. 2002] models, and value-based approach.

The VBSQM integrates the cost estimating relationships (CER's) from the Constructive Cost Model COCOMO II; the software quality attributes estimating relationships (QER's) from the Constructive Quality Model COQUALMO; and the value estimating relationships (VER's) supplied by the system's stakeholders. An initial set of cost estimating relationships (CER's) is provided by the COCOMO II model. The COCOMO II CER's enable users to express time-phased information processing capabilities in terms of equivalent software size, and to estimate time-phased software life cycle investment costs in terms of software size and the project's product, platform, people, and project attributes. An initial set of software quality attribute estimating relationships (QER's) is provided by the COQUALMO model. As an extension of the COCOMO model, COQUALMO enables users to specify time-phased levels of investment in improving dependability attributes, and to estimate the resulting time-phased dependability attribute levels. The current version of COQUALMO estimates delivered defect density in terms of a defect introduction model estimating the rates at which software requirements, design, and code defects are introduced, and a subsequent defect removal model. The relationship between COCOMO II and COQUALMO is based on the fact that the COQUALMO rating scales for levels of investment in defect removal via automated analysis, peer reviews, and execution testing and tools have been aligned with the COCOMO II RELY rating levels. The VBSQM needs initial software quality VERs supplied by the system's stakeholders. VBSQM VERs assume that stakeholders have performed a baseline business-case analysis for various components of value (profit, customer satisfaction, on-time performance) as a function of the time-phased information-processing capabilities at baseline software quality attribute levels.

The VBSQM integrating the COCOMO II, COQUALMO and VERs provides two usage scenarios to support software quality analysis from the stakeholder/value perspectives. *Scenario1 Software Quality ROI Analysis*: The integrated VBSQM framework can help project stakeholders and/or decision-makers to quantitatively determine an appropriate software quality level for a particular software project, project scenario class or software class. *Scenario 2 Combined Risk Analyses*: the framework of VBSQM, which integrates the empirically-calibrated COCOMO II and COQUALMO results and quantified stakeholder-supplied VERs , also provides the basis for us to perform combined risk analyses in order to solve the problem of how much software assurance is enough [Huang-Boehm 2005a]. We have extended the initial VBSQM discussed in [Boehm-Huang 2004a] to support such combined risk analyses.

3.4 Value Based Software Quality Achievemnt (VBSQA) Process

We propose a Value-Based Software Quality Achievement Process generated from the WinWin Spiral Model's risk-driven approach. It is coupled with a set of value-based software quality analysis methods and models for reasoning about software and system quality. It helps project success-critical stakeholders to define, negotiate and develop mission-specific combinations of software quality attributes for the development of a system with the stakeholder WinWin-balanced software quality outcome.

The entry to the VBSDA process is identified of Top-level software quality objectives with top-level mission objectives. Besides the project budget estimation, we have to perform stakeholder/value dependency analysis [Boehm-Huang 2004b] in order to understand the nature of the software quality. The Results Chain technique, developed by the DMR Consulting Group [Thorp 1998] is a way to identify missing initiatives and success-critical stakeholders in a system development project. VBSQA process framework covers all the phases and milestones in the entire software development life cycle of the WinWin Spiral model [Boehm-Hansenzz, 2001]. It also includes various software development activities to incorporate the value-based consideration. In real-world software projects, different software quality assessment criteria are set based on different business cases [Reifer 2002] so that different

process strategies should be selected to meet them. Therefore, a flexible process generation platform is required to enable the trim or addition of the steps/milestones/activities in the VBSQA process framework. Along these lines, the risk-based process decision-making approach [Huang et. al. 2006a], uses the project business case and risk analysis to tailor the VBSQA process into an overall software development strategy [Boehm-Turner 2004]. This approach relies heavily on project key stakeholder identification, project business case analysis and the collaboration of the core development team and the project stakeholders. In general, schedule-driven processes are lightweight processes that employ short iterative cycles while product-driven processes employ longer iterative cycles.

A scenario-based approach is proposed to identify stakeholders' value propositions on software quality (Q-) attributes and help stakeholders define the detailed Q-attribute requirements for different scenarios. This approach also helps to identify and resolve value conflicts on Q-attributes and reconcile form tradeoff analysis on Q-attributes in order to arrive at a mutual stakeholder WinWin- balanced software Q-attribute requirements..

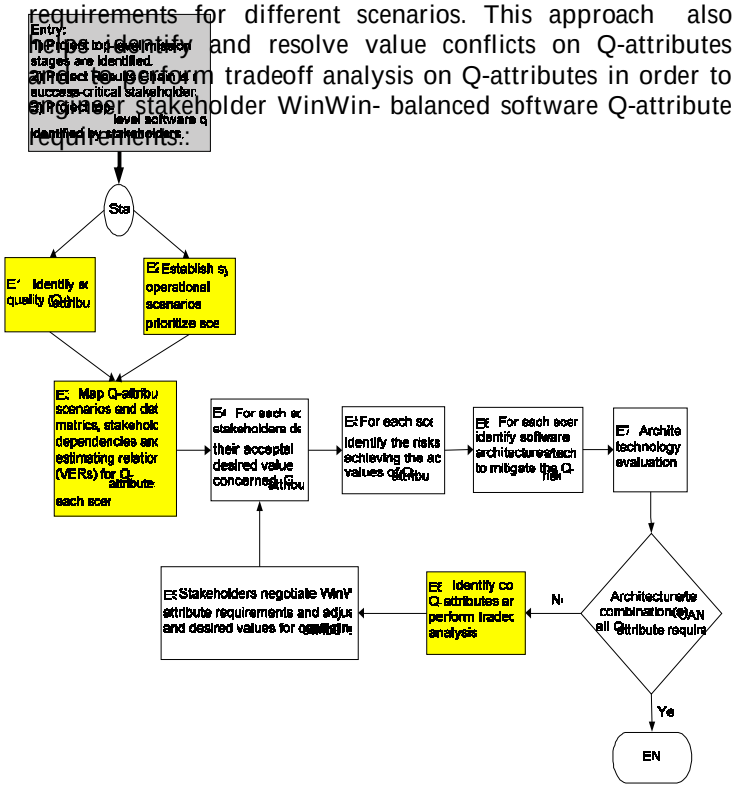


Figure1: A scenario-based approach proposed to identify stakeholders' value propositions on software quality (Q-) attributes

E1.Identify software quality (Q-) attributes; *E2*. Establish system operational profile scenarios and prioritize scenarios; *E3*. Map Q-attributes into scenarios and determine metrics, stakeholder/value dependencies and value estimating relationships (VERs) for Q-attributes of each scenario; *E4*. For each scenario, stakeholders define their acceptable and desired values for concerned Q-attributes; *E5*.For each scenario, identify the risks of not achieving the acceptable values of Q-attributes; *E6*. For each scenario, identify software architectures/technologies to mitigate the Q-risks; E7 .Architecture/technology evaluation; *E8*.Identify conflicting Q-attributes and perform tradeoff analysis; *E9*. Stakeholders negotiate WinWin balanced Q-attribute requirements and adjust the acceptable and desired values for conflicting Q-attributes.

Top-level design of at least one architecture option should be provided by developers. And the initial Feasibility Rationale Description (FRD) [MBASE 2003] furnishes the rationale for the product being able to satisfy the stakeholders' system requirements and specifications including the Q-attribute requirements. An LCO Review is to be held with the participation of all the project key stakeholders. This indicates a milestone of the LCO phase in the WinWin Spiral Model. The exit criteria of LCO ARB are to provide at least one feasible architecture to satisfy the requirements, and to provide proofs of requirement satisfaction including the Q-attribute requirements. System detailed design needs to be developed for only one feasible architecture. And the LCA Feasibility Rationale Description (FRD) [MBASE 2003] has to provide the detailed rationale of all requirement satisfaction including the Q- attribute requirements. An LCA Review is to be held with the participation of all the project key stakeholders. LCA FRD risk analysis should propose a detailed risk mitigation plan to resolve all known risks. The exit criterion of Life Cycle Architecture (LCA) Review is to commit one architecture to satisfy all the requirements of the system. The result of the LCO review is either *Pass* or *Fail*.

During the construction phase, we still need to monitor the progress of the software Q-attribute achievement and perform corrective actions when needed. We can use the mission operational scenarios and Q-attribute requirement levels defined by project key stakeholders as progress metrics and test cases. A matrix with the capability to track the value-based expected versus actual outcomes (e.g., software quality investments, reduced value loss, ROI) [Boehm- Huang 2003] is a useful technique to support the monitoring and control of the actual progress of the software quality achievement.

The Release Readiness Review (RRR) is held with the participation of all the success- critical stakeholders. If the RRR is passed, developers will perform the "cold turkey" transition of the software system to other stakeholders. If it fails, developers may be required to adjust or fix the product based on RRR feedback. Otherwise, the project will be announced as a failure.

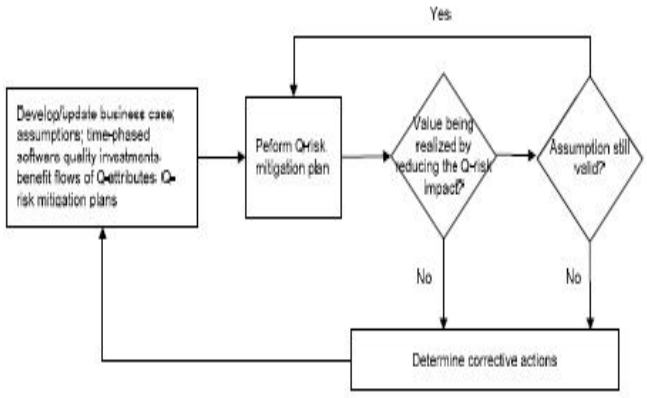


Figure 2: A Value-Realization Feedback Process to Monitor and Control the Achievement of Software Q-attribute Requirements

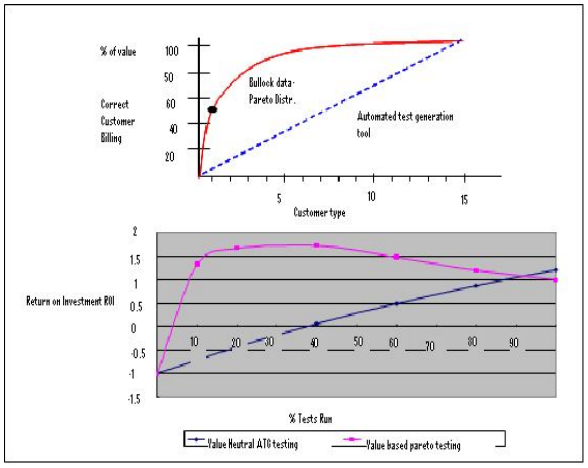


Figure 3: (a) Value Estimating Relationships (VERs) for Value-Neutral Testing vs. Value-Based Testing; (b) Return on Investment (ROI): Value- Neutral ATG Testing vs. Value-Based Pareto Testing

4 VALUE BASED TESTING VS VALUE NEUTRAL TESTINGS

4.1 Value Estimating Relationships (VERs) and ROI Analysis

With value-neutral testing, such as using the output of most automated test generation (ATG) tools, the earned mission value with invested testing effort will be linear shown as the dotted line. The value earned by each requirement will more likely follow a Pareto distribution shown as a solid curve.

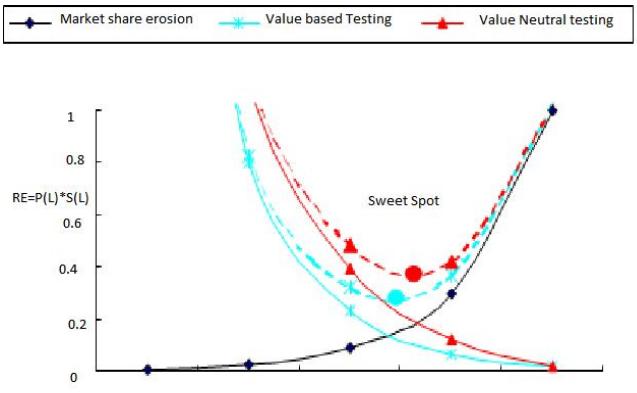
As an example from Bullock's project experience [Bullock 2000], the Return on Investment (ROI) analysis is based on the following *assumptions*: $1M of the development costs have been invested in the customer billing system by the beginning of testing, The ATG tool will cost $300K and will reduce test costs by 50% as promised, The business case for the system will produce $4M in business value in return for the $2M investment cost, The business case will provide a similar 80:20 distribution for the remaining.

% of Test Ru	Value-neutral ATG Testing				Value-based Pareto Testing			
	Cost	Value	Nt Value	ROI	Cost	Value	Nt Value	ROI
0	1300	0	-1300	-0.10	1000	0	-1000	-1.0
1	1350	400	-	-0.70	1100	2560	1460	+1.33
2	1400	800	-	-0.43	1200	3200	2000	1.67
4	1500	1600	1	+0.07	1400	3840	2440	1.74

Table 1. Comparative Business Cases: ATG and Pareto Testing

4.2 VBSQM Combined Risk Analyses

VBSQM combined risk analyses can also be applied in comparing the value- based quality achievement techniques with value-neutral ones in terms of the combined project risks. Figure 4 presents the VBSQM combined risk analysis results of value-based testing and value-neutral testing, when the high finance business case is used as an example. The decrease in *Sq(L)* with testing time will be linear for the value-neutral testing, while the decrease in *Sq(L)* with testing time will follow the negative Pareto distribution for the value-based one as shown in rows 3 and 4 at the bottom of Figure 4. The combined risk exposure of value-based testing is shown as the dashed line of triangles, while the combined risk exposure of value- neutral testing is shown as the dashed line of stars in Figure 4.4. The sweet spot of value-neutral testing moves to up and right of that of value-based testing, which is also shown in Figure 4. For this example, the minimum risk exposure for value- neutral testing is about 40% higher than that of value-based testing.



	VL	L	N	H	VH	RELY
COCOMO II	0	12	22	34	54	Add %test time

COQUALMO	1.0	.475	.24	.125	.06	Pq(L)
Value Based	3	1.68	.96	.54	.3	Sq(L): Pareto
Value Neutral	3	2.33	1.65	.975	.3	Sq(L): Linear
Market Risk	0.008	0.27	0.09	.30	1	REm

Figure 4: High Finance Combined Risk Exposures: Comparing Value-Based Testing vs. Value-Neutral Testing

With such value-based software quality models, users can perform sensitivity analyses of the most appropriate quality investment level and strategies with respect to uncertainties in stakeholder value propositions or marketplace conditions, for different risk exposure situations, or for additional qualitative considerations. The combined risk analysis model realized in VBSQM is also valuable for determining the relative payoff of value-based vs. value-neutral testing.

Even with only approximate information on relative values, the models can provide a framework to help reason about quality investment tradeoffs and decisions.atters.

5 CONCLUSION

Despite the emergent of a large number of software quality improving techniques, in practice people tend to use the value-neutral approaches in software quality analysis and achievement. The reason is that different systems have different success-critical stakeholders, and even for the same system these stakeholders may depend on it in different ways. There are no universal one-size-fits-all software quality metrics to optimize and we need to balance stakeholders different value propositions on software quality attributes.

This paper proposes a Value-Based Software Quality Analysis framework, which consists of the definitions, metrics, model, and process to address various aspects of software and system quality analysis and achievement using value-based approaches. The stakeholder/value-based definitions and metrics of software attributes differ from the traditional value-neutral ones in that they explicitly reflect the relevant success-critical stakeholders' value propositions and operational scenarios of a software system.

The VBSQM provide a technique for reasoning about the ROI of software quality attributes and performing combined risk analyses of both quality and market share erosion. It helps project decision-makers determine how much software quality investment is enough based on their project's business case. The Value-Based Software Quality Achievement (VBSQA) process, driven by the VBSQM and scenario-based approach, can be applied to determine whether a software system with stakeholder mutually satisfactory quality attribute requirements is achievable and to realize achievable stakeholder mutually satisfactory project outcomes. In spite of the practical difficulties in applying a new process in software industry where traditional Waterfall processes and methods dominated, the application experience of VBSQA process in real-world ERP software development shows that the application of value-based approaches is inherently better than the value-neutral ones that most ERP software projects have employed.

REFERENCES

[1] [Pardee 1996] W. Pardee, To Satisfy & Delight Your Customer: How to Manage for Customer Value, Dorset House Publishing, NY, 1996

[2] [Boehm et. al., 1995] B. Boehm, P. Bose, E. Horowitz, and M. Lee, "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach", Proceedings of the 17th International Conference on Software Engineering (ICSE-17), IEEE Computer Society Press, Seattle, April 1995.

[3] [IEEE 1990] ANSI/IEEE Std. 610.12-1990, IEEE Standard. Glossary of Software

[4] [Leveson 1995] N. G. Leveson, Safeware, System Safety and Computers, Addison- Wesley Publishing Company, 1995R. Nicole, "The Last Word on Decision Theory," *J. Computer Vision,* submitted for publication. (Pending publication)

[5] [IFIP WG10.4] International Federation for Information Processing (IFIP WG-10.4), www.dependability.org

[6] [Boehm et. al. 1978] B. Boehm, J. Brown, H. Kaspar, M. Lipow, G. MacLeod, M. Merritt, "Characteristics of Software Quality", TRW Report to National Bureau of Standards, November 1973; TRW Software Series Report, TRW-SS-73-09. Also published by North Holland

[7] [Gilb 1976] T. Gilb, Software Metrics, Studentlitteratur AB, Lund, Sweden, 1976.

[8] [Gilb 1988] T. Gilb, Principles of Software Engineering Management, Addison Wesley, 1988

[9] [Rus et. al. 2003] I. Rus, S. Komi-Servio, P. Costa, "Software Dependability Properties: A Survey of Definitions, Measures and Techniques," Fraunhofer Technical Report 03-110, January 2003.

[10] [LiGuo Huang 2006] LiGuo Huang, "Ph.D. Dissertation: Software Quality Analysis: A Value-Based Approach "USC-CSE, 2006.

[11] [Lyu 1996] Jean-Claude Laprie and Karama Kanoun, in Michael Lyu, ed., Handbook of Software Reliability Engineering, IEEE Computer Society Press, McGraw Hill, 1996, 27-69.

[12] [Wilson 2002] D. Wilson, B. Murphy and L. Spainhower, "Progress on Defining Standardized Classes for Comparing the Dependability of Computer Systems", Workshop on Dependability Benchmarking, in conjunction with the International Conference on Dependable Systems and Networks (DSN-2002), Washington , DC, June 2002.

[13] [Arlat 2001] J. Arlat, "Dependability Benchmarking: The SIG Class/Factor/Criteria Framework," presentation at the 39th Meeting of IFIP WG 10.4, February 2001, Paraty, Brazil.

[14] [Huynh 2003] D. Huynh, M. Zelkowitz, V. Basili and I. Rus, "Modeling Dependability for a Diverse Set of Stakeholders", the International Conference on Dependable Systems and Networks, 2003 (DSN-2003).

[15] [Shaw 2002] M. Shaw, "Everyday Dependability for Everyday needs", Keynote on 13th International Symposium on Software Reliability Engineering (ISSRE), Nov. 2002.

[16] [Reifer 2002] D. Reifer, Making the Software Business Case, Addison Wesley, 2002.

[17] [Nejmeh 2002] B. Nejmeh, I. Thomas, "Business-Driven Product Planning Using Feature Vectors and Increments", Software, November-December 2002, pp. 34-42.

[18] [Butler 2002] S. Butler, "Security Attribute Evaluation Method: A Cost-Benefit Approach", Proceedings ICSE 2002, pp. 232-240.

[19] [Aurum et. al. 2005] A. Aurum, S. Biffl, B. Boehm, H. Erdogmus, and P. Gruenbacher (eds.), Value-Based Software Engineering, Springer Verlag, 2005.

[20] [Emam 2003] Khaled El Emam, the ROI from Software Quality: An Executive Briefing, K Sharp Technology Inc., 2003.

[21] [Boehm et. al. 2000a] B. Boehm, C. Abts, A.W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Riefer, and B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[22] [Boehm-Huang 2004a] B. Boehm, L. Huang, A. Jain and R. Madachy, "The ROI of Software Dependability: The iDAVE Model", IEEE Software, vol. 21, no. 3, May/June 2004, pp.54-61.

[23] [Huang-Boehm 2005a] L. Huang and B. Boehm, "Determining How Much Software Assurance Is Enough? A Value-based Approach", Proceeding of the 7th International Workshop on Economics-Driven Software Engineering Research (EDSER), May 2005.

[24] [Boehm-Huang 2004b] B. Boehm, L. Huang, A. Jain, and R. Madachy, "The Nature of Information System Dependability: A Stakeholder/Value Approach" (Draft 6)", USC-CSE Technical Report, December, 2004.

[25] [Steece, et. al. 2002] B. Steese, S. Chulani, B. Boehm, " Determining Software Quality Using COQUALMO," Case Studies in Reliability and Maintenance, W. Blischke and D. Murthy, eds., Jon Wiley & Sons, 2002.

[26] [Thorp 1998] J. Thorp and DMR, The Information Paradox, McGraw Hill, 1998.

[27] [Boehm-Hansenzz, 2001] B. Boehm, W. Hansenzz, "Understanding the Spiral Model as a Tool for Evolutionary Acquisition", CrossTalk, May, 2001.

[28] [Boehm-Turner 2004] B. Boehm, R. Turner, Balancing Agility and Discipline, Addison Wesley, 2004.