Value-Based Software Metrics

Barry Boehm, Winsor Brown
USC, Los Angeles, CA, USA
(boehm; awbrown)@cse.usc.edu

Abstract

Many software metrics, particularly dependability metrics, are usually presented in a value-neutral framework. For example, the primary metric for Reliability, Mean Time Between Failures, is referenced to a one-size-fits-all (and often not explicitly defined) definition of "failure." This can lead to serious problems, as when operators' optimization on liveness produces user response times to queries of up to 2 weeks (an actual example). Clearly, unacceptable response time is a "failure" for some stakeholders that should be reflected in the metric definition. The paper presents some initial thoughts and a candidate approach for addressing this issue.

1. Current Interests

USC's Center for Software Engineering has been working on value-based software engineering for several years; see next section.

2. Past Work

Our past work has included the book *Software Engineering Economics(Boehm)*, the book *Software Cost Estimation with COCOMO II (Boehm and Brown, 2 co-authors)*, and the stakeholder win-win and benefits realization approaches to benefits analysis.

3. Issue Statement

Applying one-size-fits-all value-neutral approaches in software engineering is a frequent source of problems. This issue came up in the NASA High Dependability Computing Program, which needed a framework of dependability definitions that would enable NASA to make good decisions about deploying limited resources to achieve the most valuable combination of various dependability metrics.

4. Proposed Approach

We have been experimenting with a framework of stakeholder value-based dependability definitions to enable decisionmakers to relate their dependability decisions to their particular set of key stakeholders and values.

Previous work

The software quality metrics field has generally tried to relate its metrics definitions to the uses that the software provided (performance, dependability, adaptability, usability, etc.); see references [1]-[6]. Still, as seen in [7], there is not much consensus on the definition of software quality and how to achieve it.

5. Results, Status, Prospects, and Needs

A critical objective of the NASA High Dependability Computing Program is a definition of "dependability" that enables the program to evaluate the contributions of existing and new computing technologies to the improvement of an information-intensive system's dependability. Such evaluations require one or more evaluation criteria or metrics that enable quantitative comparisons of candidate technology solutions to be performed.

Ideally, one would like to have a single Dependability metric by which the contributions of each technology could be ranked. 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.

Thus, a critical first step in understanding the nature of information system dependability is to identify the major classes of success-critical information system stakeholders, and to characterize the relative strengths of their dependencies on various attributes of a given information system. This involves answering three main questions:

1. What are the primary attributes of an information system that success-critical stakeholders depend on?
2. What classes of information system stakeholders exhibit different patterns of dependency on these attributes?
3. For each class of stakeholder, what is the relative strength of their dependency on each attribute?

Table 1 provides a top-level summary of the relative strengths of dependency on information system dependability attributes, for classes of information system stakeholders exhibiting different patterns of strengths of dependency. Its initial portions were obtained from empirical analysis of different classes of information system stakeholders' primary concerns during win-win requirements negotiations.

Most of the terms are reasonably self-explanatory, but two may require further explanations. System dependents are people that are not involved in the system's development or operation, but are dependent on some of its attributes (e.g., safety for airline passengers and medical patients). System controllers perform real-time control of a system (e.g., airline pilots or electric power distribution controllers). Stakeholders may belong to multiple classes: an airline pilot is both a system controller and a system-dependent passenger.

Table 1. Top-Level Stakeholder/Value Dependencies

** Critical * Significant () Insignificant or indirect — Stakeholder Classes / Dependability Attributes	Info Suppliers	System Dependents	Info Brokers	Information Consumers: Mission-critical	Information Consumers: uncrit.	System Controllers	Developers	Maintainers	Administrators	Acquirers
Protection										
Safety		**		**		**				
Security	*	**	**	**		**				
Privacy	**		*	*	*					
Robustness										
Reliability			*	**		**		*	*	
Availability			*	**		**		*	*	
Survivability		**	*	**		**				
Quality of Service										
Performance				**	*	**		*	*	
Accuracy, Consistency	**		**	**	*	**		*	*	
Accessibility, ease of use;	*		*	**	**	**			*	
Difficulty of misuse										
Evolvability			*	*	*	*		**		**
Interoperability			**						*	**
Correctness							**			**
Cost							*			**
Schedule			*	*	*	*	**			**
Reusability							**	*		*

The dependency ratings refer only to direct dependencies. For example, system developers, acquirers, controllers and administrators are concerned with safety or security only to the extent that a system's information suppliers, users, and dependents are concerned with them. And information suppliers and system dependents are only concerned with reliability and availability to the extent that these help provide their direct concerns with security and safety.

Challenges in Applying the Stakeholder/Value Dependency Framework

Again ideally, one would like to apply the framework in Table 1 to identify the relative success-criticality of each class of stakeholder for a given (kind of) system; determine the relative strength of each stakeholder class's dependency on each dependability attribute; and derive a set of attribute weights that could be combined into a single Dependability metric that could be used as the criterion for evaluating candidate approaches for developing this (kind of) system.

Again, however, in practice, some additional challenges arise:

- Individuals in each stakeholder class may not have the same dependency strengths, and some individuals may be more success-critical than others.
- Strengths of dependency will vary by operational context or mission scenario (e.g., normal vs. crisis performance).
- Strengths of dependency will also vary by Maslow need hierarchy level, in which unsatisfied lower-level needs dominate (having enough Availability to stay in business will dominate having strong Security), but in which satisfied lower-level needs are no longer motivators (having the ultimate in Availability in a viable business will become less important than having more Security.)
- The dependability attributes are not neatly orthogonal or independent. Sometimes they reinforce each other, and sometimes they conflict with each other.
- The relationships between what some technologies do (e.g., remove certain classes of defects) and what stakeholders depend on (e.g., achieve desired levels of Reliability or Availability) are often not straightforward.

Capabilities Provided by the Stakeholder/Value Dependency Framework

Even with its complexities and challenges, the stakeholder/value dependency framework provides

some much-needed capabilities for reasoning about dependability attributes in technology assessment or project scoping. These capabilities include:

- It corroborates the statement above that there is no universal one-size-fits-all dependability metric that one can optimize.
- It emphasizes the importance of including all of a system's success-critical stakeholders in prioritizing dependability attributes.
- It provides first-order guidance on which stakeholder classes to consult in determining a system's dependability-attribute priorities
- It explicitly identifies sources of complexity in dependability assessment, and helps avoid the measurement dysfunction accompanying overly simplistic dependability improvement initiatives.
- Along with complementary analyses of dependability-attribute complementarities and conflicts, it supports tradeoff analyses among dependability attributes.
- It highlights the importance of using operationally representative stakeholders and scenarios in evaluating a system's dependability attributes.
- It provides the basis for developing specific processes for evaluating a candidate dependability strategy for project use, or for evaluating a candidate technology's ability to enhance a system's dependability attributes.

The stakeholder/value dependency framework is a work in progress; a more detailed report elaborates on the nature of the stakeholders, attributes, and dependencies. Portions are based on empirical analysis; other portions are currently hypotheses awaiting empirical test. It thus provides both a working approach for empirical assessment of a system's dependability attributes, and for empirical research in validating and strengthening the assessment framework.

6. Open Issues

As just mentioned, the stakeholder/value dependency framework is a work in progress, and could use some review and discussion at EDSER-6.

7. References

[1] Boehm, B., Brown, J. & Lipow, M: Quantitative Evaluation of Software Quality, Proceedings of the Second International Conference on Software Engineering, 1976, pp592-605.

[2] McCall, J., Richards, P. & Walters, G.: Factors in Software Quality, Vol 1,2, & 3, November 1977.

[3] Kitchenham, B. & Walker, J.: The Meaning of Quality, *Software Engineering 86: Proceedings of BCS-IEE Software Engineering 86 Conference*, Southampton England September 1986.

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

[5] Wong, B. & Jeffery, R.: Cognitive Structures of Software Evaluation: A Means-End Chain Analysis of Quality, *Proceedings of the Third International Conference, PROFES 2001*, 2001,pp 6-26.

[6] Chulani, S., Santhanam P., Moore D., Leszkowicz B., Davidson G., "Deriving a Software Quality View from Customer Satisfaction and Service Data', *European Software Conference on Metrics and Measurement*, Mar 2001.3.

[7] Eickelmann, N. & Hayes, J., New Year's Resolutions for Software Quality, *IEEE Software*, pp.12-13, Jan./Feb. 2004.

8. Biography

Barry Boehm is the TRW Professor of Software Engineering and Director of the Center for Software Engineeing at the University of Southern California. His contributions to the field include the Constructive Cost Model (COCOMO), the Spiral Model of the software process, the Theory W (win-win) approach to software management and requirements determination.

Winsor Brown is the Assistant Director of the Center for Software Engineering at the University of Southern California. His 30 years in the software field include extensive experience in the commercial and aerospace industries. He has made numerous contributions to USC's value-based software engineering approach, particularly in the software quality area.