

# Reasoning About the Value of Dependability: The iDAVE Model

Barry Boehm  
Department of Computer Science  
University of Southern California  
Los Angeles, CA 90089, USA  
(213)-740-8163  
boehm@sunset.usc.edu

LiGuo Huang  
Department of Computer Science  
University of Southern California  
Los Angeles, CA 90089, USA  
(213)-740-6505  
liguohua@usc.edu

Apurva Jain  
Department of Computer Science  
University of Southern California  
Los Angeles, CA 90089, USA  
(213)-740-6505  
apurvaja@usc.edu

## ABSTRACT

In this paper, we present a framework for reasoning about the value of information processing dependability investments called the Information Dependability Attribute Value Enhancement (iDAVE) model. We describe the overall structure of iDAVE, and illustrate its use in determining the ROI of investments in dependability for a commercial order processing system. We conclude that dynamic and adaptive value-based dependability mechanisms such as iDAVE model will become increasingly important provided evidence that dependability attribute requirement levels tend to be more emergent than pre-specifiable.

## General Terms

Measurement, Design, Economics.

## Keywords

Value, Cost, Dependability, Return On Investment

## 1. INTRODUCTION

This paper presents a framework for reasoning about the value of information processing dependability investments called the Information Dependability Attribute Value Enhancement (iDAVE) model. It assumes that a project makes baseline investments in developing information processing capabilities that generate baseline flows of costs, benefits, and returns on investment (ROI). From this baseline, iDAVE provides ways to specify additional investments in enhancing dependability, and to determine the resulting additional costs and the resulting improvements in both dependability attribute levels and their ensuing system benefits. These can then be used to determine the ROI of the dependability investments.

The paper describes the overall structure of iDAVE, and illustrates its use in determining the ROI of investments in dependability for a commercial order processing system. It shows how the results not only provide a useful decision aid for dependability investments, but also provide deeper insights on the nature of dependability investment processes, and on the nature of information processing dependability analysis methods.

## 2. NATRUE AND STRUCTURE OF THE iDAVE MODEL

The overall form of the iDAVE model is shown in Figure 1. An initial set of cost estimating relationships (CER's) is provided by the COCOMO II model [3]. The COCOMO II CER's enable users to express time-phased information processing capabilities in terms of equivalent size, and to estimate time-phased investment costs in terms of size and the project's product, platform, people, and project attributes. Additional future CER's would include the COCOTS CER's for COTS-related software costs [1], inventory-based CER's for hardware components and COTS licenses, and activity-based CER's for associated investments in training and business process re-engineering.

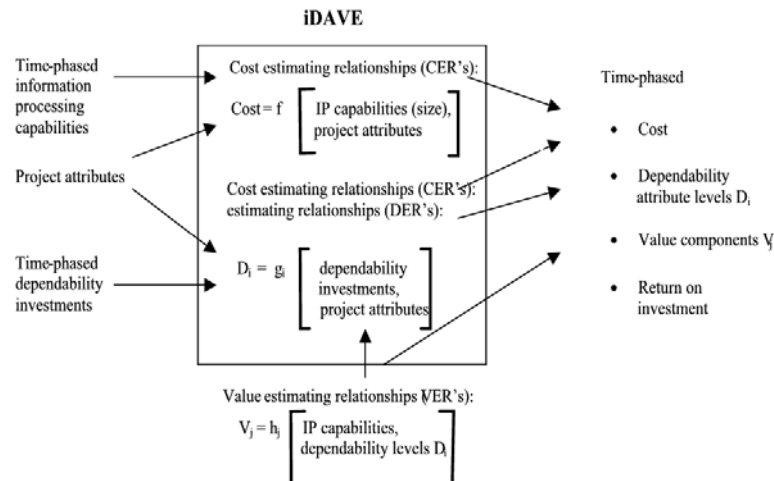


Figure 1. Proposed Information Dependability Attribute Value Enhancement (iDAVE) Model

An initial set of dependability attribute estimating relationships (DER's) is provided by the COQUALMO model [7]. It 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 defect introduction rates are determined as a function of calibrated baseline rates modified by multipliers determined from the project's COCOMO II product, platform, people, and project attribute ratings. The defect removal model estimates the rates of defect removal as a function of the project's levels of investment in automated analysis tools, peer reviews, and execution testing and tools. Initial CER's are available to estimate the costs of these investments. Further COQUALMO extensions will refine its current DER's, and will provide further DER's for estimation of additional dependability attributes such as reliability, availability, and security [5].

The iDAVE model's initial dependability value estimating relationships (VER's) assume that a baseline business case analysis has been performed for various components of value (profit, customer satisfaction, on-time performance) as a function of the time-phased information processing capabilities at nominal dependability attribute levels. These value components are aggregated into an overall time-phased value stream, which is then composed with the time-phased costs (cost of IP capabilities plus dependability investments) and normalized using present-value formulas to produce a time-phased return on investment profile.

The initial iDAVE VER's involved simple relationships such as the operational cost savings per delivered defect avoided, or the loss in sales per percent of the system downtime. Future extensions are planned to involve more detailed dependability VER's for defect severity distributions and reliability/availability levels, and additional VER's for such dependability attributes as security risk profiles and safety hazard profiles.

### 3. AN INITIAL iDAVE PROOF-OF-PRINCIPLE ANALYSIS: DEPENDABLE ORDER PROCESSING

The example below illustrates a simple initial use of iDAVE to develop a rough dependability return on investment analysis, using the Sierra Mountainbikes order processing system business case analysis in [2]. It uses this business case analysis as the baseline for assessing future investments in dependability over and above the nominal investments usually made for business data processing systems. Table 1 summarizes the business case for an improved order processing system through its proposed development in 2004-2005 and proposed operation in 2005-2008.

More specifically, the Initial Operational Capability (IOC) for the order processing system will start development on January 1, 2004. It will be installed for beta-testing with the three key distributors on September 30, 2004, and cut over as a replacement for most of the old system on December 31, 2004, at a cumulative investment cost of \$4 million. An incremental release of the IOC responding to the most cost-effective fixes and enhancements will occur on March 31, 2005. Concurrently, work will start on the enhancements for the Full Operational Capability (FOC), which will also be beta-tested by the three key distributors, and then cut

over as a full replacement for the old system on December 31, 2005, at a cumulative cost of \$6 million. Thereafter, six-month increments and annual new releases will be installed at an annual investment level of 500K.

Table 1 shows the corresponding expected benefits and return on investment,  $ROI = (Benefits - Costs) / Costs$ , annually for the years 2004-2008. For simplicity in this analysis, the costs and benefits are shown in 2004 dollars to avoid the complications of discounted cash flow calculations, and the 10% annual growth rate in estimated market size is not compounded, both for simplicity and conservatism.

As seen in columns 2-5 of Table 1, Sierra's current market share and profit margins are estimated to stay roughly constant over the 2004-2008 period, with annual profits growing from \$7M to \$12M, if the new program is not executed. This is a conservative estimate, as the problems with the current system would increase with added sales volume, leading to decreased market share and profitability.

The next columns in Table 1 up through ROI show the expected improvements in market share and profit margins (due both to economies of scale and decreased operational costs) achievable with the new system, and the resulting ROI relative to continuing with the current system. They show that the expected increase in market share (from 20% to 30% by 2008) and profit margins have produced a 45% ROI by the end of the second year of new-system operation (2006):

$$ROI = \frac{Benefits - Costs}{Costs} = \frac{9.4 - 6.5}{6.5} = 0.45$$

The expected ROI by the end of 2008 is 297%.

The final four columns in Table 1 show expected 2004-2008 improvement in overall customer satisfaction and three of its critical components: percentage of late deliveries, ease of use, and in-transit visibility. The latter capability was identified as both important to distributors (if they know what is happening with a delayed shipment, they can improvise workarounds), and one which some of Sierra's competitors were providing. Sierra's expected 2004-2008 improvements with the new system were to improve their 0-5 satisfaction rate on in-transit visibility from a low 1.0 to a high 4.6, and to increase their overall customer satisfaction rate for order processing from 1.7 to 4.6.

### 4. iDAVE DEPENDABILITY ROI ANALYSIS AND RESULTS

The iDAVE dependability ROI analysis begins by analyzing the effect of increasing dependability investments from the normal business levels to the next higher levels of investment in analysis tool support (\$260K), peer review practices (\$210K), and test thoroughness (\$314K). These correspond to the Nominal and High COQUALMO rating scale levels in Table 2. The resulting total investment of \$784K yields COQUALMO estimates of a decrease in delivered defect density from 15 defects per thousand lines of code (D/KSLOC) to 3 D/KSLOC, and an increase in mean time between failures (MTBF) from 300 hours to 10,000 hours.

**Table 1. Order Processing System: Expected Benefits and Business Case**

Date	Current System				New System				Cost Savings	Change in Profits	Cum. Change in Profits	Cum. Cost	ROI	Late Delivery %	Cust. Satis. 0-5	In-Tran. 0-5	Ease of Use 0-5
	Market Size (\$M)	Market Share %	Sales	Profits	Market Share %	Sales	Profits										
12/31/03	360	20	72	7	20	72	7	0	0	0	0	0	12.4	1.7	1.0	1.8	
12/31/04	400	20	80	8	20	80	8	0	0	0	4	-1	11.4	3.0	2.5	3.0	
12/31/05	440	20	88	9	22	97	10	2.2	3.2	3.2	6	-47	7.0	4.0	3.5	4.0	
12/31/06	480	20	96	10	25	120	13	3.2	6.2	9.4	6.5	45	4.0	4.3	4.0	4.3	
12/31/07	520	20	104	11	28	146	16	4.0	9.0	18.4	7	1.63	3.0	4.5	4.3	4.5	
12/31/08	560	20	112	12	30	168	19	4.4	11.4	29.8	7.5	2.97	2.5	4.6	4.6	4.6	

Assuming a mean time to repair of 3 hours yields an improvement in availability =  $MTBF/(MTBF + MTTR)$  from 300/303 ~ .99 to 10,000/10,003 ~ .9997.

If we use availability as a proxy for dependability, and assume that a 1% increase in downtime is roughly equivalent to a 1% loss in sales, we can use the Sierra Mountainbikes business case to determine a dependability Value Estimating Relationship (VER). Applying the difference between a .01 loss in sales and a .0003 loss in sales to the 2005-2008 Sierra new system sales total of \$531M (adding up the 2005-2008 numbers in column 7 of Table 1) yields a net return on the dependability investment of (.01) (\$531M) - (.0003) (\$531M) = \$5.31M - 0.16M = \$5.15M. The COCOMO II Cost Estimating Relationships (CER's) for Tool Support and Process Maturity also generate software rework savings from the investments in early defect prevention and removal of \$0.45M, for a total savings of \$5.59M. The resulting dependability ROI is  $(5.59 - 0.784) / 0.784 \sim 6:1$ . A related interesting result is that added dependability investments have relatively little payoff, as there is only \$0.16M left to be saved by decreasing downtime.

This analysis makes a number of assumptions that will require considerable added research to fully justify, but it provides a proof of principle that the COCOMO II CER's, the COQUALMORDER's, and business-case based VER's can be used to produce reasonable estimates of the high-payoff and lower-payoff regions for investments in dependability.

**Table 2. Defect Removal Investment Rating Scales**

Rating	Automated Analysis	Peer Reviews	Execution Testing and Tools
<b>Very Low</b>	Simple compiler syntax checking.	No peer review.	No testing.
<b>Low</b>	Basic compiler capabilities for static module-level code analysis, syntax, type-checking.	Ad-hoc informal walkthroughs Minimal preparation, no follow-up.	Ad-hoc testing and debugging. Basic text-based debugger
<b>Nominal</b>	Some compiler extensions for static module and inter-module level code analysis, syntax, type-checking. Basic requirements and design consistency, traceability checking.	Well-defined sequence of preparation, review, minimal follow-up. Informal review roles and procedures.	Basic unit test, integration test, system test process. Basic test data management, problem tracking support. Test criteria based on checklists.
<b>High</b>	Intermediate-level module and inter-module code syntax and semantic analysis. Simple requirements/design view consistency checking.	Formal review roles with all participants well-trained and procedures applied to all products using basic checklists, follow up.	Well-defined test sequence tailored to organization (acceptance / alpha / beta / flight / etc.) test. Basic test coverage tools, test support system. Basic test process management.
<b>Very High</b>	More elaborate requirements/design view consistency checking. Basic distributed-processing and temporal analysis, model checking, symbolic execution.	Formal review roles with all participants well-trained and procedures applied to all product artifacts & changes (formal change control boards). Basic review checklists, root cause analysis. Formal follow-up. Use of historical data on inspection rate, preparation rate, fault density.	More advanced test tools, test data preparation, basic test oracle support, distributed monitoring and analysis, assertion checking. Metrics-based test process management.
<b>Extra High</b>	Formalized* specification and verification. Advanced distributed processing and temporal analysis, model checking, symbolic execution.  *Consistency-checkable pre-conditions and post-conditions, but not mathematical theorems.	Formal review roles and procedures for fixes, change control. Extensive review checklists, root cause analysis. Continuous review process improvement. User/Customer involvement, Statistical Process Control.	Highly advanced tools for test oracles, distributed monitoring and analysis, assertion checking. Integration of automated analysis and test tools. Model-based test process management.

## 5. CASE STUDY IMPLICATIONS

More importantly, though, the case study's ability to relate a specific dependability analysis to specific human decisionmaking issues opens the door to an entirely new set of speculations and research directions about the nature of value-based decisionmaking and its implications for dependability-oriented software engineering. In the case study, because added investments in availability have little payoff, does this mean that the Sierra decisionmakers will lose interest in further

dependability investments? Probably not. More likely, they are operating within a Maslow need hierarchy in which satisfied availability needs are no longer motivators, but in which higher-level needs such as reducing security risks may now become more significant motivators.

This casts the analysis of dependability attributes in an entirely new light. Previously, the problem of software attribute analysis has been largely cast as an exercise in static multi-attribute optimizing or satisficing, operating on some pre-weighted combinations of dependability attribute satisfaction levels. The practical decision making issue above indicates that achieving an acceptable or preferred combination of dependability attributes is generally not a pre-specifiable problem but rather a dynamic process in which satisfaction of currently top-priority dependability attributes leads to a new situation in which the attribute priorities are likely to change.

In this situation, dependability attribute requirements become more emergent than pre-specifiable. The process for achieving acceptable dependability becomes no longer a single-pass process, but an evolutionary process, subject to the need to anticipate and develop architectural support for downstream dependability needs. The types of dependability analyzers that become important increasingly involve the types of dynamic and adaptive value-oriented dependability mechanisms being explored in papers such as [4, 6, 8].

## 6. CONCLUSIONS

The multidimensional nature of “dependability” decisionmaking is compounded by the potentially very high levels of investment required to achieve very high levels of dependability. This creates a demand for methods of reasoning about the cost and value of achieving various levels of dependability attributes in particular project situations.

The iDAVE model presented here provides an overall framework and an initial set of tools for reasoning about the value of dependability. Application of an initial version of iDAVE to an example project decisionmaking situation shows that the model can produce reasonable estimates that distinguish higher-payoff and lower-payoff regions of a project’s investment in dependability.

Use of the iDAVE model in this decision situation provided evidence that dependability attribute requirement levels tend to be more emergent than pre-specifiable; that dependability analysis and achievement processes tend to be more evolutionary than

single-pass; and that dynamic and adaptive value-based dependability mechanisms will become increasingly important.

## 7. REFERENCES

- [1] Abts, C., Boehm, B., and Clark, E., "COCOTS: A Software COTS-Based System (CBS) Cost Model," Proceedings ESCOM, 2001.
- [2] Boehm, B. and L. Huang, “Value-Based Software Engineering: A Case Study,” to appear, *IEEE Software*, March 2003.
- [3] 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.
- [4] David Garlan and Bradley Schmerl, "Model-based Adaptation for Self-Healing Systems," ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02), November 18-19, 2002.
- [5] Reifer, D., Boehm, B., and Gangadharan, M., "Estimating the Cost of Security for COTS Software," Proceedings Second Intl. Conf. COTS-Based Software Systems, February 2003.
- [6] Mary Shaw, “Self-Healing’: Softening Precision to Avoid Brittleness,” Proceedings of the First ACM SIGSOFT Workshop on Self-Healing Systems (WOSS '02) Charleston, South Carolina, November 2002, pp. 111-113.
- [7] Steece, B., Chulani, S., and Boehm, B., "Determining Software Quality Using COQUALMO," in *Case Studies in Reliability and Maintenance*, W. Blischke and D. Murthy, Eds.: Wiley, 2002
- [8] K. Sullivan and J. Knight, “Information Survivability Control Systems,” Proceedings of the 21<sup>st</sup> International Conference on Software Engineering, May, 1999, pp. 184—193.