The ROI of Software Dependability: The iDAVE Model

Barry Boehm, LiGuo Huang, Apurva Jain, and Ray Madachy,
University of Southern California

In most organizations, proposed investments in software dependability compete for limited resources with proposed investments in software and system functionality, response time, adaptability, speed of development, ease of use, and other system capabilities. The lack of good return-on-investment models for software dependability makes determining the overall business case for dependability investments difficult. So, with a weak business case, investments in software dependability and the resulting system dependability are frequently inadequate.

Because different stakeholders depend on different system capabilities (such as availability, safety, or security) in different situations, the business case for dependability must deal with multiple situation-dependent attribute values. Dependability models will need to support stakeholders in determining their desired levels for each dependability attribute and estimating the cost, value, and ROI for achieving those. At the University of Southern California, researchers have developed software cost- and quality-estimation models[1,2] and value-based software engineering processes, methods, and tools.[3,4] (For other work in this area, see the "Related Work" sidebar.) We used these models and the value-based approach to develop an Information Dependability Attribute Value Estimation model for reasoning about software dependability's ROI.

Weak business cases often lead to inadequate investments in software dependability and in the resulting system dependability. The iDAVE model estimates the relative return on investment for achieving desired dependability attribute values. It also helps select the most effective software dependability strategies.

The iDAVE model

Figure 1 shows iDAVE's overall structure, which includes *cost-estimating relationships* from the Constructive Cost Model COCOMO II.[1] CERs let users express time-phased information-processing capabilities in terms of equivalent software size. They also let users estimate time-phased software life-cycle investment

Related Work

Researchers have developed, with encouraging results, general frameworks for reasoning about software quality attributes[1,2] and specific frameworks that define dependability attributes.[3,4]

At the Economics-Driven Software Engineering Research workshops (www.edser.org) and elsewhere,[5,6] researchers have developed general frameworks for making software engineering decisions that enhance the value of delivered software systems. In addition to work at the University of Southern California, other EDSER contributions that explicitly address dependability aspects include

- Carnegie Mellon University's work on value-based security investment analysis,[7] warranty models for software,[8] and value-based software fault detection[9]
- The University of Virginia's application of real-options theory to the value of modularity[10] and application of utility-theory and stochastic control approaches to reliable delivery of computational services[11]

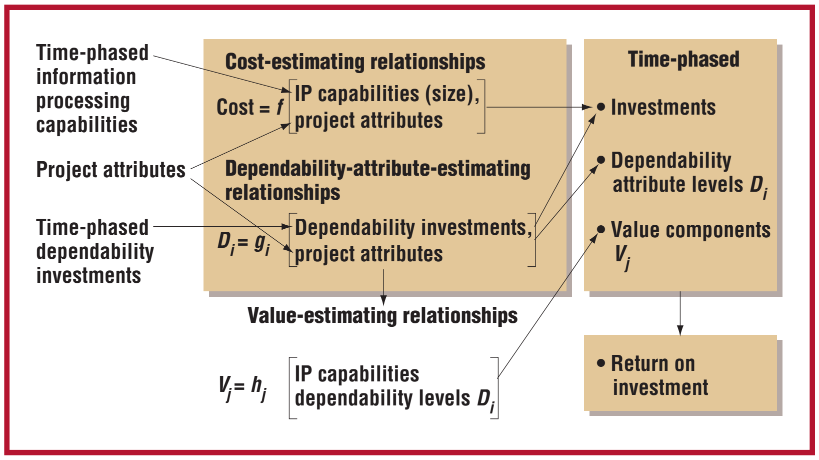
References

1. L. Chung et al., *Non-Functional Requirements in Software Engineering*, Kluwer, 1999.
2. B. Boehm et al., *Characteristics of Software Quality*, tech. report TRW-73-09, 1973 (also North Holland, 1978).
3. A. Avezienis, J. Laprie, and B. Randell, "Fundamental Concepts of Computer System Dependability," *IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments*, 2001; www.cs.virginia.edu/~jck/cs651/papers/laprie.taxonomy.pdf.
4. I. Rus, S. Komi-Servio, and P. Costa, *Software Dependability Properties: A Survey of Definitions, Measures and Techniques*, tech. report 03-110, Fraunhofer Center for Experimental Software Eng., 2003.
5. D. Reifer, *Making the Software Business Case*, Addison-Wesley, 2002.
6. B. Nejmeh and I. Thomas, "Business-Driven Product Planning Using Feature Vectors and Increments," *IEEE Software*, Nov./Dec. 2002, pp. 34–42.
7. S. Butler, "Security Attribute Evaluation Method: A Cost-Benefit Approach," *Proc. 24th Int'l Conf. Software Eng.* (ICSE 02), IEEE CS Press, 2002, pp. 232–240.
8. P. Li et al., "The Potential for Synergy between Certification and Insurance," *Special Edition of ACM SIGSOFT from the 1st Int'l Workshop Software Reuse Economics* (in conjunction with the 7th Int'l Conf. Software Reuse), 2002; www.sei.cmu.edu/staff/kcw/icsr02.pdf.
9. O. Raz and M. Shaw, "Software Risk Management and Insurance," *Proc. 3rd Int'l Workshop Economics-Driven Software Eng. Research*, IEEE CS Press, 2001; www.cs.virginia.edu/~sullivan/edser3/raz.pdf.
10. K. Sullivan et al., "Software Design as an Investment Activity: A Real Options Perspective," *Real Options and Business Strategy: Applications to Decision Making*, L. Trigeorgis, ed., Risk Books, 1999.
11. Y. Cai and K. Sullivan, "Stochastic Optimal Switching," *Proc. 4th Workshop on Economics-Driven Software Eng. Research*, IEEE CS Press, 2002, pp. 71–72.

costs in terms of software size and the project's product, platform, people, and project attributes. Eventually, iDAVE might also include CERs for COTS-related software costs, inventory-based CERs for hardware components and COTS licenses, and activity-based CERs for associated investments in training and business process reengineering.

iDAVE also includes an initial set of *dependability-attribute-estimating relationships* from the Constructive Quality Model COQUALMO.[5] DERs let users specify time-phased levels of investment in improving dependability attributes, from which COQUALMO estimates the resulting time-phased dependability attribute levels. The current version of COQUALMO estimates delivered-defect density in terms of a defect introduction model—which estimates the rates at which software requirements, design, and code defects are introduced—and in terms of a subsequent defect-removal model. COQUALMO determines the rates at which software requirements, design, and code defects are introduced as a function of calibrated baseline rates modified by multipliers determined from the project's COCOMO II product, platform, people, and project attribute ratings. For example, a Very Low rating for applications experience will lead to a significant increase in requirements defects but only a small increase in code defects.

The defect removal model estimates removal rates as a function of the project's levels of investment in automated analysis tools, peer reviews, and execution testing and tools (see Table 1). Initial CERs are available to estimate the costs of these investments. Further COQUALMO extensions will refine current



Time-phased information processing capabilities
Project attributes
Time-phased dependability investments
Cost-estimating relationships
Cost = f [IP capabilities (size), project attributes]
Dependability-attribute-estimating relationships
$D_i = g_i$ [Dependability investments, project attributes]
Value-estimating relationships
$V_j = h_j$ [IP capabilities, dependability levels D_i]
Time-phased
• Investments
• Dependability attribute levels D_i
• Value components V_j
• Return on investment

Figure 1. The proposed Information Dependability Attribute Value Estimation (iDAVE) model.

Table 1

COQUALMO defect-removal-investment rating scales

Rating	Automated analysis	Peer reviews	Execution testing and tools
Very Low	Simple compiler syntax checking	No peer review	No testing
Low	Basic compiler capabilities	Ad-hoc informal walkthroughs	Ad-hoc testing and debugging
Nominal	Compiler extension Basic requirements and design consistency	Well-defined sequence of preparation, review, and minimal follow-up	Basic test, test data management, and problem-tracking support Test criteria based on checklists
High	Intermediate-level module and intermodule Simple requirements and design	Formal review roles with well-trained participants using basic checklists and follow-up procedures	Well-defined test sequences tailored to the organization Basic test coverage tools, test support system Basic test process management
Very High	More elaborate requirements and design Basic distributed-processing and temporal analysis, model checking, and symbolic execution	Basic review checklists and root cause analysis Formal follow-up using historical data on inspection rate, preparation rate, and fault density	More advanced test tools and test data preparation, basic test oracle support and distributed monitoring and analysis, and assertion checking Metrics-based test process management
Extra High	Formalized specification and verification Advanced distributed processing	Formal review of roles and procedures Extensive review checklists and root cause analysis Continuous review-process improvement Statistical process control	Highly advanced tools for test oracles, distributed monitoring and analysis, and assertion checking Integration of automated analysis and test tools Model-based test process management

DERs and add new ones for estimating additional dependability attributes such as reliability, availability, and security.[6] For example, one extension will offer more precise treatment of "bad fixes," which average 7 percent of all defect fixes and over 10 percent of defect-fixing efforts. The current COQUALMO defect introduction model is calibrated to the total number of defects introduced, including bad fixes. This is a reasonable first approximation, although it's insensitive to the defect removal rate.

On the basis of the framework just discussed, iDAVE has this usage scenario:

1. Use a checklist of dependability attributes to involve stakeholders in prioritizing attributes of highest concern and usage scenarios.
2. Estimate software size in terms of value-adding capabilities.
3. Enter the size and baseline cost drivers into COCOMO II to obtain baseline cost estimates.
4. Enter baseline and alternative dependability drivers into COCOMO II and COQUALMO, and obtain alternative cost and dependability estimates.
5. Involve stakeholders in determining the appropriate form and parameters for value-estimation relationships (discussed later).
6. Apply iDAVE to assess the costs, benefits, and ROIs for the alternatives.
7. Iterate the previous steps as appropriate.

An initial iDAVE spreadsheet tool is available at http://sunset.usc.edu/cse/pub/research/iDAVE/iDAVE.zip.

Relations between COCOMO II and COQUALMO

For reliability, the COCOMO II Required Reliability (RELY) CER provides an initial bridge to dependability estimation, expressed in terms of the operational impact of software defects (see Table 2). Participants at a USC industry-government Affiliates' Workshop translated this bridge into a rough experience-based DER for software reliability in terms of mean time between failure (MTBF) in hours (see Table 2). Table 2 also shows the corresponding effort multipliers (relative levels of effort or cost) to achieve the associated reliability levels, as calibrated from experience data on 161 diverse software projects. For example, developing software for users with low, easily recoverable losses (such as PC users) results in an MTBF of 10 hours (roughly a daily crash) and a relative cost of 0.92. Developing software for financial organizations, where crashes can cause high financial losses, results in an MTBF of 10K

Table 2

COCOMO II required reliability rating scale, mean time between failure (MTBF), and relative cost

	RELY rating				
	Very Low	**Low**	**Nominal**	**High**	**Very High**
Definition	Slight inconvenience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life
MTBF (hrs.)	1	10	300	10K	300K
Relative cost	0.82	0.92	1.0	1.10	1.26

hours (417 days or somewhat over a year between crashes) and a relative cost of 1.10.

We've aligned the COQUALMO rating scales in Table 1 with the COCOMO II RELY ratings. We can thus compare the levels of investment for the Low and High COCOMO II rating levels with the tools and activities used at these levels in the COQUALMO rating scales. To cover the COQUALMO Extra High rating level in Table 2, we have provisionally extended the reliability rating scale in Table 1 to Extra High, with a corresponding MTBF of 1M (million) hours and a relative cost of 1.56. This is based on some experiences with thorough independent verification and validation efforts, which added about 30 percent to software costs.

For ratings between Very Low and Extra High, iDAVE provides two ways to apply this relationship between COCOMO II and COQUALMO. One is to specify a COCOMO II RELY rating and assume the same investment levels in automated analysis, peer reviews, and execution testing will be applied, in which case the corresponding relative effort and MTBF will be used. Or, we can specify our own investment levels for automated analysis, peer reviews, and execution testing and tools, using a COQUALMO-based weighted average of these levels as the equivalent COCOMO II RELY rating. We could use other cost models such as Knowledge Plan, PRICE S, SEER, and SLIM in place of COCOMO II, to the extent that they have a similarly defined RELY CER.

Specifying value-estimating relationships

The iDAVE model needs initial dependability *value-estimating relationships*, supplied by the system's stakeholders, to relate estimated cost investments and dependability levels to resulting benefit flows and ROI estimates. iDAVE VERs assume that stakeholders have performed a baseline business-case analysis for various components of value (profit, customer satisfaction, and ontime performance) as a function of the time-phased information-processing capabilities at baseline dependability attribute levels. iDAVE aggregates these value components into an overall time-phased value stream, comprising the time-phased costs (the cost of IP capabilities plus dependability investments) and normalizing them using present-value formulas to produce a time-phased ROI profile.

The initial iDAVE VERs involve simple relationships such as the operational cost savings per delivered defect avoided, or the loss in sales per percent of system downtime, shown as the linear VERs in Figure 2. Many organizations providing e-services also use such relationships to measure loss of revenue due to system downtime. For example, on the higher side, Intel estimates its loss of revenue as $275K ($US) for every hour of order-processing-system downtime; other companies estimate $167K (Cisco), $83K (Dell), $28K (Amazon), and $3K (Ebay).[7] Another typical value-versus-availability relationship can appear as a pro-

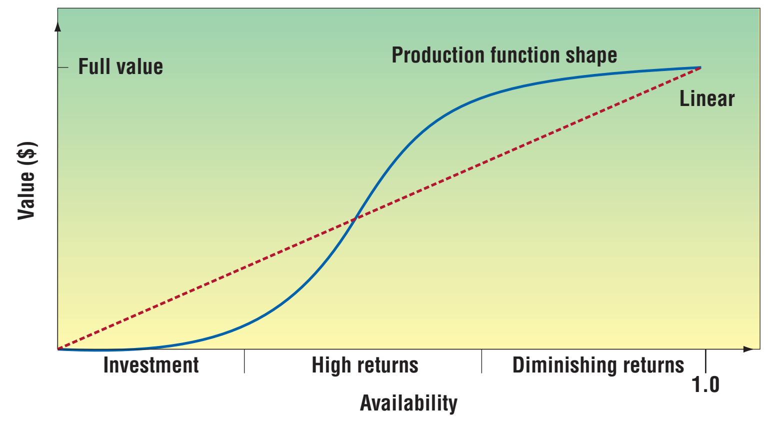


Figure 2. Two typical relationships between realized value and availability.

Table 3

The current order-processing system

Year	Market size ($M)	Market share (%)	Sales ($M)	Profits ($M)
2003	360	20	72	7
2004	400	20	80	8
2005	440	20	88	9
2006	480	20	96	10
2007	520	20	104	11
2008	560	20	112	12

duction function (see Figure 2). Below a certain level of investment, as with the Ashton-Tate DBase-4 bankruptcy, incremental gains in availability don't avoid bankruptcy. Beyond this level, there's a *high-returns segment*, but at some point, incremental investments in availability don't affect users' frustration levels, resulting in a *diminishing-returns segment*.

ROI analysis of an order-processing system

Here we use iDAVE to develop a rough dependability ROI analysis, using the Sierra Mountainbikes order-processing-system business case analysis (a case study representative of two recent order-processing projects).[6] iDAVE uses this business case analysis as the baseline for assessing future investments in dependability beyond the nominal investments usually made for business data-processing systems.

An initial proof-of-principle analysis

We estimate that with Sierra's current system, its market share and profit margins would remain roughly constant between 2004 and 2008, with annual profits growing from $7M to $12M (see Table 3). This is a conservative estimate, because problems with the current system would increase with added sales volume, leading to decreased market share and profitability.

Table 4 summarizes the business case for an improved order-processing system through its proposed development in 2004–2005 and operation in 2005–2008. The cumulative cost of fully replacing the old system with the new one is $6M, of which $3.44M is for software.

Table 4 shows the expected improvements in market share and profit margins (due to economies of scale and decreased operational costs) achievable with the new system, and the resulting ROI relative to continuing with the current system. The expected increase in market share (from 20 to 30 percent by 2008) and profit margins should produce a 45 percent 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 percent. (For simplicity, we show the costs and benefits in 2004 dollars to avoid the complications of discounted-cash-flow calculations. Additionally, the 10 percent annual growth rate in estimated market size isn't compounded.)

Table 4 also shows that we must estimate and track qualitative and quantitative ROI aspects. It shows a simple balanced scorecard array of expected 2004–2008 improvements in overall customer satisfaction and three of its critical components: percentage of late deliveries, ease of use, and in-transit visibility. This last capability is both important to distributors (because if they know what is happening with a delayed shipment, they can improvise

Table 4

The new order-processing system's expected benefits and business case

Year	Market share (%)	Sales ($M)	Profits ($M)	Cost savings ($M)	Change in profits ($M)	Cumulative change in profits ($M)	Cumulative cost ($M)	ROI	Late delivery (%)	Customer satisfaction (0–5)	In-transit visibility (0–5)	Ease of use (0–5)
2003	20	72	7	0	0	0	0	0	12.4	1.7	1.0	1.8
2004	20	80	8	0	0	0	4	–1	11.4	3.0	2.5	3.0
2005	22	97	10	2.2	3.2	3.2	6	–.47	7.0	4.0	3.5	4.0
2006	25	120	13	3.2	6.2	9.4	6.5	.45	4.0	4.3	4.0	4.3
2007	28	146	16	4.0	9.0	18.4	7	1.63	3.0	4.5	4.3	4.5
2008	30	168	19	4.4	11.4	29.8	7.5	2.97	2.5	4.6	4.6	4.6

Table 5

ROI analysis results of the Sierra Mountainbikes' order-processing system and NASA's planetary rover (increasing MTBF)

Project	RELY rating	MTBF (hrs.)	MTTR (hrs.)	Availability	Loss ($M)	Increased value ($M)	Cost ($M)	Change ($M)	ROI
Sierra	Nominal	300	3	0.9901	5.31	0	3.45	0	—
Mountainbikes	High	10K	3	0.9997	0.16	5.15	3.79	0.344	14.0
System	Very High	300K	3	0.99999	0.005	0.155	4.34	0.55	–0.72
	Extra High	1M	3	1	0	0.005	5.38	1.04	–1.0
NASA	High	10K	150	0.9852	4.44	0	22	0	—
Planetary	Very High	300K	150	0.9995	0.15	4.29	25.2	3.2	0.32
Rover	Extra High	1M	150	0.99985	0.045	0.105	31.2	6.0	–0.98

workarounds) and as a capability that some Sierra competitors provide. Sierra expects the new system to improve its 0–5 satisfaction rate on in-transit visibility from a low 1.0 to a high 4.6, and to increase its overall customer satisfaction rate for order processing from 1.7 to 4.6.

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, peer-review practices, and test thoroughness. The resulting increase from a Nominal to High COCOMO II RELY rating increases the MTBF from 300 to 10K hours. It also incurs an additional $344K ($3.44M (1.10 – 1.0)) in software dependability investments.

We measure availability as MTBF/(MTBF + MTTR), where MTTR is the mean time to repair. If we assume (from relevant business experience) that the MTTR is three hours, the initial availability of approximately .99 (300/303) improves to approximately .9997 (10,000/10,003). If we use availability as a proxy for dependability and assume that a 1 percent increase in downtime is roughly equivalent to a 1 percent loss in sales, we can use the Sierra Mountainbikes business case to determine a dependability VER. If we apply the difference between a .01 loss in sales and a .0003 loss in sales to Sierra's new system, then a sales total of $531M (adding up the 2005–2008 sales numbers in Table 4) yields a net return on the dependability investment of

(.01) ($531M) – (.0003) ($531M) =
$5.31M – 160K = $5.15M.

The resulting dependability ROI is (5.15 – 0.344) / 0.344 = 14.0.

A related result is that added dependability investments have relatively little payoff, because the company can only save $0.16M by decreasing downtime. Table 5 summarizes the iDAVE ROI analysis results by increasing the dependability level of the Sierra Mountainbikes order-processing system from Nominal to High, High to Very High, and Very High to Extra High.

We discussed this analysis in nondirected interviews with relevant business personnel (including a representative of one of the failed projects on which we based the Sierra Mountainbikes case study). In the interviews, the personnel indicated that the results were realistic and that using iDAVE would have helped them in several decision situations.

However, some respondents indicated that their interest in dependability wouldn't disappear with the negative ROI (shown in Table 5) that resulted from going from a High to a Very High RELY level. They said other concerns such as security would likely deserve further investment once availability stopped causing significant business losses. This suggests that organizations such as Sierra Mountainbikes operate 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 might become more significant motivators once basic business viability is achieved.

This casts the analysis of dependability attributes in a new light. Previously, the problem of software attribute analysis was largely cast as an exercise in static multiattribute optimizing or satisficing, operating on some preweighted combinations of dependability at-

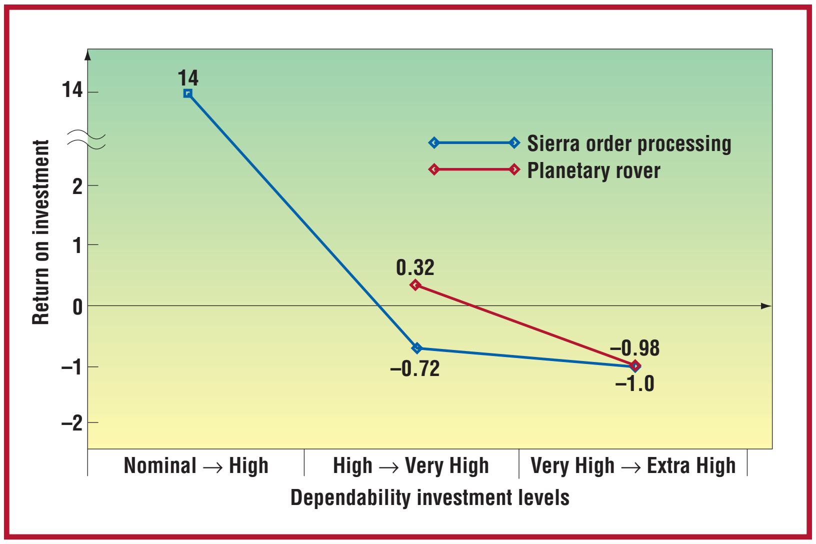


Figure 3. Comparison of iDAVE ROI analysis results for the Sierra Mountainbikes order-processing system and NASA's planetary rover.

tribute satisfaction levels. The practical decision-making issue surfacing here indicated that achieving an acceptable or preferred combination of dependability attributes leads to a new situation in which the attribute priorities will likely change.

In this situation, dependability attribute requirements become more emergent than prespecifiable. The process for achieving acceptable dependability is then no longer a single-pass process. Rather, it is an evolutionary process, subject to the need to anticipate and develop architectural support for downstream dependability needs.

A planetary rover's dependability

As part of the NASA High-Dependability Computing Program, and in collaboration with Jet Propulsion Laboratory (JPL) Mission Data System and Mars Science Laboratory personnel, we also performed an exploratory iDAVE analysis of a representative NASA planetary-rover robot. The personnel indicated that a planetary rover's top-priority dependability attribute is survivability. Otherwise, its first failure on a remote planet will be its last.

Survivability has certain attributes—such as graceful degradation—so that the rover can at least keep enough power and communication capability to transmit its status to its Mission Control Center on Earth. It should also be able to receive and execute recovery commands from the Mission Control Center. However, because availability is strongly correlated with survivability and is more straightforward to analyze, we used availability as a proxy for survivability.

We'd need a more detailed hazard analysis and fault-tolerance and graceful-degradation cost-benefit analysis for safety or survivability ROI analyses. Additionally, we'd need a more detailed vulnerability and protection cost-benefit analysis for security ROI analysis.

As part of a business case for determining planetary-rover dependability VERs, we used a total mission value equal to a representative planetary-rover mission cost of $300M, with a baseline software cost of $20M at a Nominal COCOMO II software RELY rating. We also assumed as a baseline that a 1 percent decrease in availability was roughly equivalent to a 1 percent loss in the mission value of $300M. For a representative MTTR, the JPL personnel indicated that 150 hours or roughly a week was a representative amount of time for a Mission Control Center to diagnose a planetary rover problem, to formulate and prepare a recovery sequence, and to test its validity.

Because a planetary rover needs at least a High level of required reliability, the iDAVE dependability ROI analysis begins by analyzing the effect of increasing the dependability investment from High to Very High. This corresponds to the COCOMO II estimates of a cost increase of 16 percent from $22M to $25.2M and an increase in MTBF from 10K hours to 300K hours (see Table 2). As Table 5 shows, an assumed MTTR of 150 hours improves availability from 0.985 to 0.9995.

Further increasing the RELY rating from Very High to Extra High goes outside the COCOMO II rating-scale range and requires a special analysis or the use of the extended COCOMO II RELY range. Investing in an independent verification and validation activity to bring the MTBF up to 1M hours (114 years), using the COQUALMO Extra High levels of activity, incurs estimated additional investments in formal-analysis-tool support and usage ($2M), peer review practices ($1M), and test thoroughness ($3M). As a cross-check, the resulting $6M investment is near the usual value of 30 percent added cost for independent verification and validation on such missions. At this point, the added dependability investments have negative payoff (ROI = –0.98), because decreasing downtime will only save $0.105M. Table 5 summarizes the iDAVE ROI analysis results by increasing the planetary rover's dependability level from High to Very High and from Very High to Extra High.

Comparing ROI analysis results

Figure 3 compares the iDAVE ROI analysis results for the Sierra Mountainbikes order-processing system and the NASA planetary rover.

So we see that different mission situations lead to different diminishing-returns points for the business application, whose ROI goes negative when going from a High to a Very High RELY rating. The planetary rover application's positive ROI is sustained through Very High but not through Extra High.

When we discussed this outcome in a nondirected group discussion with the JPL personnel, they pointed out that the losses in reputation, in corrective action, and in some cases human life (as with the Columbia Shuttle failure) can amount to much more lost value than the mission's cost. Consequently, these can produce a positive ROI for an Extra High dependability effort.

Also, we've assumed a linear value model in this analysis, which results in the monotone decreasing of the mission value by decreasing the dependability level. Most often, value models look like the S-shaped economic-production functions in Figure 2, with an initial low-slope investment segment, a high-slope high-returns segment, and a final low-slope diminishing-returns segment.

JPL personnel are proposing additional research to further calibrate iDAVE to JPL experience. They also hope to experimentally use the model to evaluate the relative ROI of alternative combinations of dependability technologies to planetary-mission outcomes. Additional challenges involve developing and calibrating domain-oriented DERs and VERs for additional dependability attributes such as security, safety, accuracy, and performance assurance.

Acknowledgments

The NASA High Dependability Computing Program and the NSF Information Technology Program support iDAVE research. We also thank JPL's Dan Dvorak, Kenny Meyer, Robert Rasmussen, and Kirk Reinholtz for their insights on planetary rover dependability.

About the Authors

Barry Boehm is the TRW professor of software engineering at the University of Southern California. His primary research interest is value-based software engineering. He has a PhD in math from the University of California, Los Angeles. He is a fellow of the ACM, the AIAA (American Institute of Aeronautics and Astronautics), the IEEE, and INCOSE (the International Council on Systems Engineering). Contact him at the Univ. of Southern California, Computer Science Dept., Henry Salvatori Computer Science Center, Los Angeles, CA 90089-0781; boehm@sunset.usc.edu.

LiGuo Huang is a PhD student in the University of Southern California's Computer Science Department and a research assistant at the University's Center for Software Engineering. Her primary research interests are value-based software engineering and high dependability computing. She received her MS in computer science from USC. Contact her at the Univ. of Southern California, Computer Science Dept., Henry Salvatori Computer Science Center, Los Angeles, CA 90089-0781; liguohua@usc.edu.

Apurva Jain is a PhD student at the University of Southern California's Center for Software Engineering. His primary research interests are value-based software engineering and pervasive computing. He received his BS from Curtin University of Technology, and his professional honors diploma from Informatics, Singapore. Contact him at the Univ. of Southern California, Computer Science Dept., Henry Salvatori Computer Science Center, Los Angeles, CA 90089-0781; apurvaja@usc.edu.

Raymond Madachy is an adjunct assistant professor at the University of Southern California, a research scientist with the USC Center for Software Engineering, and CTO at the Cost Xpert Group. His research involves value-based software engineering, process improvement and modeling, cost estimation, metrics, and risk management. He earned his PhD in industrial and systems engineering from USC. He is a senior member of the IEEE and a member of INCOSE (the International Council on Systems Engineering), Tau Beta Pi, and Pi Tau Sigma. Contact him at the Univ. of Southern California, Computer Science Dept., Henry Salvatori Computer Science Center, Los Angeles, CA 90089-0781; madachy@sunset.usc.edu.

References

1. B. Boehm et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.
2. B. Steece, S. Chulani, and B. Boehm, "Determining Software Quality Using COQUALMO," *Case Studies in Reliability and Maintenance*, W. Blischke and D. Murthy, eds., John Wiley & Sons, 2002.
3. B. Boehm, "Value-Based Software Engineering," *ACM Software Eng. Notes*, vol. 28, no. 2, 2003, pp. 29.
4. B. Boehm and L. Huang, "Value-Based Software Engineering: A Case Study," *Computer*, vol. 36, no. 3, 2003, pp. 33–41.
5. S. Butler, "Security Attribute Evaluation Method: A Cost-Benefit Approach," *Proc. 24th Int'l Conf. Software Eng.* (ICSE 02), IEEE CS Press, 2002, pp. 232–240.
6. D. Reifer, B. Boehm, and M. Gangadharan, "Estimating the Cost of Security for COTS Software," *Proc. 2nd Int'l Conf. COTS-Based Software Systems*, Springer-Verlag, 2003, pp. 178–186.
7. R. DeMillo, "Why Software Falls Down," *Mutation Testing for the New Century*, W.E. Wong, ed., Kluwer Academic, 2001.