Sufficient Extent of Software Quality Investment

Meer Shizan Ali

Abstract— Determining when to stop testing and releasing the product for use is the basic problem faced by many software projects. Though we have risk analysis, which helps to address that what is the sufficient extent of software quality investment by balancing the risk exposure of doing too little with the risk exposure of doing too much. However, people have often found it difficult to quantify the relative probabilities and sizes of loss in order to provide practical approaches for determining a risk-balanced "sweet spot" operating point. Here a quantitative approach is provided based on the COCOMO II cost estimation model and the COQUALMO quality estimation model to help project decision-makers determine that what extent of software quality investment is enough. Also examples of its use under differing value profiles are provided. Further, some representative empirical data and models are used to assess the relative payoff of value-based testing as compared to value-neutral testing.

Index Terms— Software Quality, Software Risk Assessment, Software Cost/Schedule Estimation, Software Testing, Cost-Benefit Analysis, Value-Based Software Engineering, COCOMO, COQUALMO

———————————— ◆ ————————————

1 INTRODUCTION

Is quality really free, as Philip Crosby would have us believe? [7] Does a higher-quality Lexus cost less to produce and purchase than a lower-quality Corolla? Unfortunately, there are no simple, one-size-fits-all answers to such questions. Each quality investment situation must be analyzed to determine as well as possible what levels of investment are not enough and what levels are too much.

A good example in the software field is how to determine when to stop testing and release the product for use. We have found that risk analysis helps to address such "how much is enough?" questions, by balancing the risk exposure (probability of loss time's impact of loss) of doing too little with the risk exposure of doing too much. However, people have often found it difficult to quantify the relative probabilities and economic value of loss in order to provide practical approaches for determining a risk-balanced "sweet spot" operating point.

This paper draws on results from the emerging field of Value-Based Software Engineering (VBSE) to provide a quantitative approach for addressing how much software quality investment is enough, based on the well-calibrated COCOMO II cost estimation model and the partially-calibrated COQUALMO quality estimation model. We also provide examples of its use under differing value profiles characterizing early startups, routine business operations, and high-finance operations. Further, we show how the model and approach can assess the relative payoff of value-based testing as compared to value-neutral testing.

2 DEVELOPMENT COST OF "REQUIRED RELIABILITY": COCOMO II

The core of the Constructive Cost Model (COCOMO) II [1] is a mathematical relationship involving 24 variables used to estimate the amount of effort in person-months required to devel-

———————————————

• *Meer Shizan Ali, Assistant Professor MIT, Indore*
  PH-09926029535, E-mail: mshizan@gmail.com

op a software product defined by the variables. By multiplying the project effort by its cost per person-month, one can also estimate the project's cost.

COCOMO II's parameters include the product's equivalent size in thousands of lines of code (KSLOC) or a function-point equivalent; personnel characteristics such as capability, experience, and continuity; project characteristics such as execution-time and storage constraints; and product characteristics such as complexity, reusability, and required reliability. The parameters are calibrated to a diverse 161-project sample of well-measured projects. The sample may not be fully representative of software projects in general, as many projects do not measure their performance.

The regression analysis of the 161 projects produced a relative effort range of 1.54 between projects reporting their required reliability (RELY) as Very Low (the impact of a product failure was a slight inconvenience) and projects reporting a Very High RELY rating (the impact of a product failure was a risk of loss of human life). The t-value produced by the regression analysis for the RELY variable was 2.602, well above the statistical significance level of 1.96 for this sample size and number of variables [1; page169].

The added effort for a Very High RELY project is the net result of rework savings due to early error elimination and extra effort in very thorough off-nominal, model-based, stress, and regression testing at the end of the project. Since the extra effort occurs near the end, when the project is about at its average staffing level, it roughly translates into an extra 54% of calendar time in thorough testing before fielding the product.

These results are summarized in Figure 1. Based on data from a subset of the projects, we have also added a rough scale of product Mean Time Between Failures (MTBF) corresponding to the relative impact of product failures, going from 1 hour MTBF for Very Low RELY to 300K hours MTBF for Very High RELY.

Figure 1: COCOMO II Software Development Cost/ Reliability/ Test Time Tradeoff
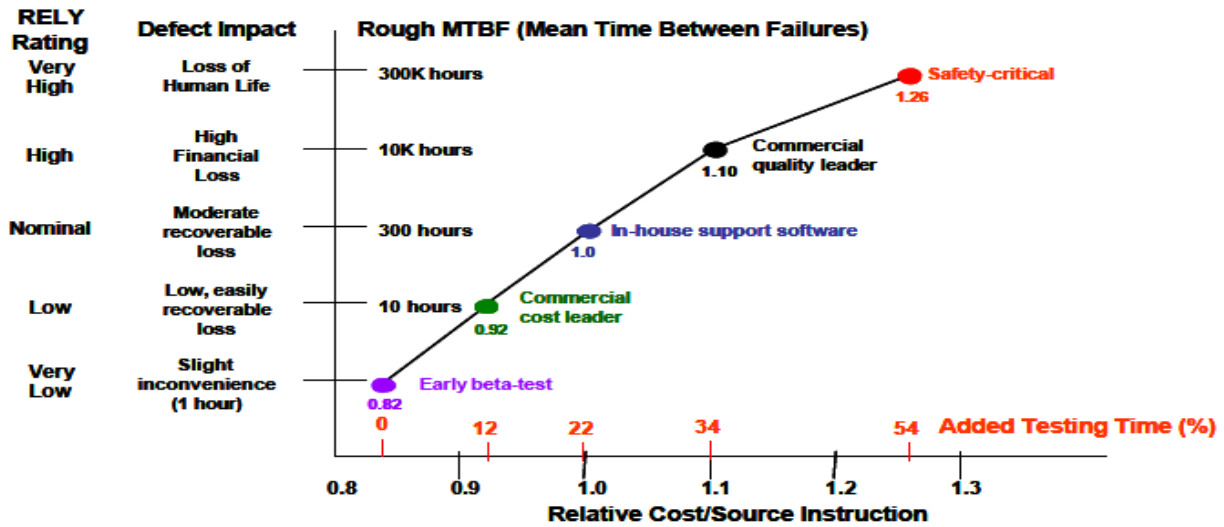
3 COST OF "REDUCED DELIVERED DEFECT DENSITY": COQUALMO

As an extension of the COCOMO model, COQUALMO [1, 2] enables users to specify time-phased levels of investment in improving software quality measured by defect densities in requirements, design and code, and to estimate the resulting time-phased reliability 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. For example, a Very Low rating for Applications Experience will lead to a significant increase in requirements defects introduced, and a smaller increase in code defects introduced. The defect removal model estimates the rates of defect removal as a function of the project's levels of investment in peer reviews, automated analysis tools, and execution testing and tools. Its rating scales range from Very Low to Extra High.

The calibrated baseline (i.e., nominal) defect introduction rates for COQUALMO are 9 requirements defects/KSLOC, 19 design defects/KSLOC, and 33 code defects/KSLOC. For simplicity and to avoid unwarranted precision, we have rounded these to 10, 20 and 30, for a total of 60 defects/KSLOC introduced [1]. Starting from this baseline, the COQUALMO estimation of reduced delivered defect density as a function of the composite defect removal rating is shown in Figure 2. The Very Low composite defect removal rating leaves delivered defect density at 60 Delivered Defects/KSLOC (DDK), while an Extra High rating can reduce the delivered defect density to only 1.6 DDK [1; page 266]. Note that the composite defect removal rating is an integration of the ratings for automated analysis tools, peer reviews, and execution testing and tools.

Note also that it assumes nominal rates of defect introduction: a strong defect prevention program can reduce delivered defect densities by another factor of 60 to 100. The RELY Cost-Estimating Relationship (CER) in COCOMO II discussed in section 2 is available to estimate the cost of these investments, as its Very Low to Very High rating levels correspond to the horizontal rows of defect reduction investments in Table 1. For mixed levels of investment in analysis, reviews, and testing, COQUALMO DDK estimates and an equivalent RELY rating can also be determined.
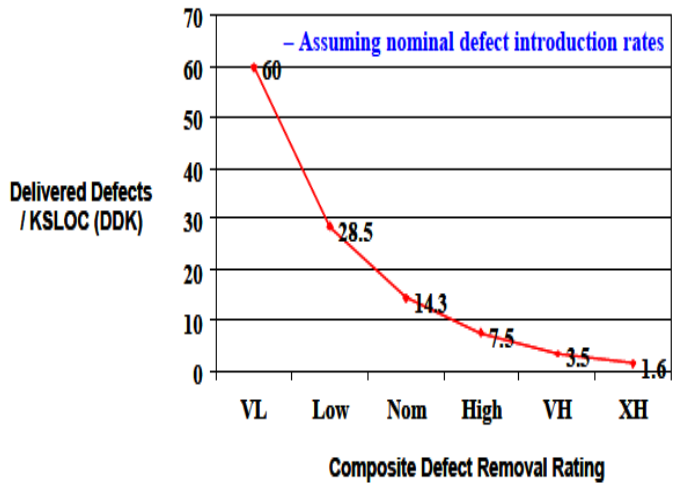

Figure 2. COQUALMO Reduced Delivered Defects Estimates at Nominal Defect Introduction Rates

Table 1. Defect Removal Investment Rating Scales   delivery.

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, minimal follow-up.	Basic test, test data management, problem tracking support. Test criteria based on checklists.
High	Intermediate-level module and inter-module; Simple requirements/design	Formal review roles with well-trained participants and using basic checklists, follow up.	Well-defined test sequence tailored to organization. Basic test coverage tools, test support system. Basic test process management.
Very High	More elaborate requirements/design Basic distributed-processing and temporal analysis, model checking, symbolic execution.	Basic review checklists, root cause analysis. Formal follow-up using 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.
Extra High	Formalized specification and verification. Advanced distributed processing	Formal review roles and procedures. Extensive review checklists, root cause analysis. Continuous review process improvement. 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.

4 RELATIONSHIP BETWEEN COCOMO II AND COQUALMO

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 shown in Figure 1. The correspondence between COCOMO II RELY ratings and COQUALMO defect removal profile ratings is based upon a mapping between the activity analysis behind the COCOMO RELY effort multiplier and the COQUALMO defect removal activity ratings. One can thus compare the levels of investment for the Low and High COCOMO II rating le vels with the tools and activities assumed to be used at these levels in the COQUALMO rating scales. This relationship between CO-COMO II and COQUALMO also produces a way to relate investments in software reliability to resulting values of the delivered system's Mean Time between Failures (MTBF), as shown in Figure 1. An example of the use of this relation to determine "how much availability is enough" for various classes of applications is provided in [3].

5 VALUE ESTIMATING RELATIONSHIPS

Finally, we need value estimating relationships (VERs) supplied by project critical stakeholders to relate software quality levels or product delivery time to resulting benefit flows and value earned. VERs assumes that stakeholders have performed a baseline business case analysis for various components of value (profit, customer satisfaction, and on-time performance, etc.) as a function of the software quality investment (i.e., peer review, automated analysis and automated analysis, and execution testing and tools) levels. In the e-service domain, the major VERs involves losses in market share due to insufficient software quality and/or delayed product

6 SUFFICIENCY OF SOFTWARE QUALITY INVESTMENT USING COCOMO II, COQUALMO AND VER'S

6.1 Combined Risk Analysis

In Figure 3, we will build up information from COCOMO II, COQUALMO, and the VERs discussed above to show what level of quality investment is enough for various situations. The probability of loss $P_q(L)$ (e.g., financial, reputation, future prospects) due to unacceptably low quality can be estimated based on the COQUALMO estimate of delivered defect density in Figure 2: to first order, the fewer the defects, the lower the probability of loss. We can use the Very Low estimate of 60 defects/KSLOC in Figure 2 as the baseline for $P_q(L)$, and set its default value to 1. The $P_q(L)$ for other RELY ratings from Low to Very High can then be computed based on the corresponding delivered defect density relative to the baseline, as shown in the second row of numbers at the bottom of Figure 3. A baseline VER for the size of loss $S_q(L)$ due to unacceptable quality can be obtained for value-based testing [4,5,6] from the Pareto distribution, using a negative Pareto distribution for value loss as shown in rows 3, 4 and 5 at the bottom of Figure 3. In Figure 3, relative $S_q(L)$ is shown in three representative business cases such as early start-up (row 3), representing relatively defect-tolerant early adopters; normal commercial (row 4), representing the Bullock data; and high finance (row 5), representing very high-volume time-sensitive cash flows dependent on reliable operation of the software system. For simplicity, we use a factor of 3 to distinguish the relative values of the three cases. Then we can compute the software quality investment

Risk exposure as RE $_q$ = $P_q(L)$ x $S_q(L)$

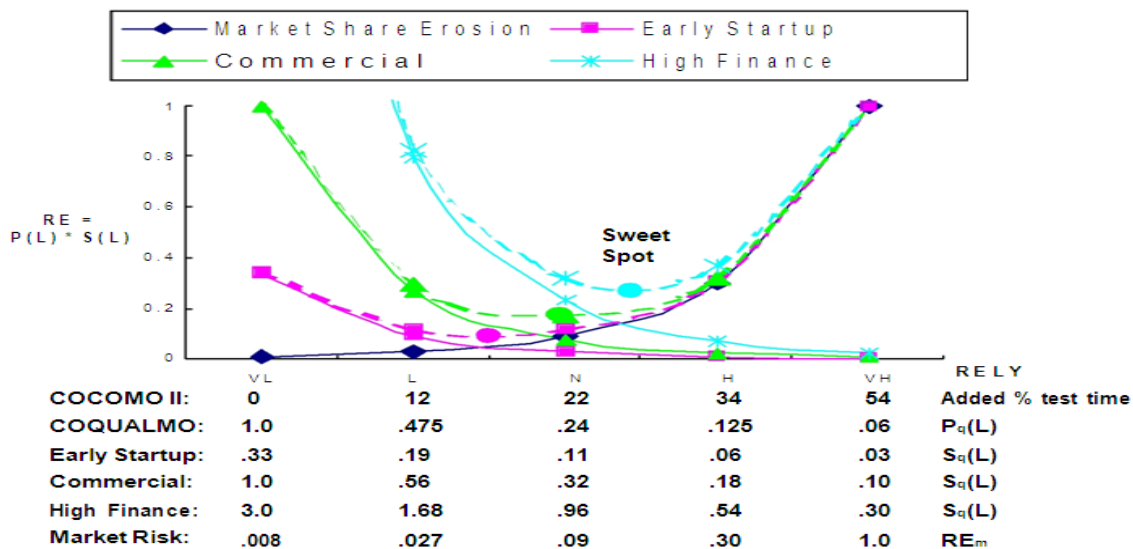|	VL	L	N	H	VH	
COCOMO II:	0	12	22	34	54	Added % test time
COQUALMO:	1.0	.475	.24	.125	.06	$P_q(L)$
Early Startup:	.33	.19	.11	.06	.03	$S_q(L)$
Commercial:	1.0	.56	.32	.18	.10	$S_q(L)$
High Finance:	3.0	1.68	.96	.54	.30	$S_q(L)$
Market Risk:	.008	.027	.09	.30	1.0	RE_m

Figure 3. Combined Risk Exposures: Early Startup, Commercial and High Finance

These values enable us to calculate relative quality investment risk exposures as functions of added testing time for the three classes of business cases. Each of these classes of stakeholders can then determine their own "how much software quality investment is enough?" sweet spot by combining their software quality investment risk exposure curve with their market share erosion risk exposure curve RE_m shown as the line of diamonds in Figure 3. For simplicity, we have shown this to be equal to 1.0 for a Very High RELY rating and an added CO-COMO -calibrated 54% delay in time to market, and decreasing by a factor of 0.3 for each successively lower RELY rating, as shown in the bottom line of numbers in Figure 3.

Finally, we can find a sweet spot (the minimum) from the combined risk exposure due to both unacceptable software quality and market share erosion. Figure 3 shows the three combined RE curves in dashed lines and the corresponding oval sweet spots of software quality investment levels for the three business cases. For the high finance business case, its sweet spot of software quality investment is located at the right-most side because the risk exposure of low system quality RE_q dominates. For the early startup business case, its sweet spot of software quality investment located at the left-most side because the risk exposure of high market share erosion RE_m dominates. Such risk analyses can help project decision-makers determine where is the optimal stopping point in planning for "how much testing will be enough," or more generally, the optimal level of the software quality investment for their project based on their own business case.

The base lining at 1.0 of the highest mainstream size of loss due to low software quality and of the highest risk exposure due to market share erosion means that the model in Figure 3 can be straightforwardly adapted to other business situations. For example, a software vendor in the High Finance market sector could replace the 1.0 baseline market share risk exposure with his/her estimate of a $10M loss in late delivery of a

new feature in row 6 of Figure 3 by multiplying the numbers in row 6 by $10M. Similarly, he/she could adjust the numbers in row 4 by replacing the 1.0 baseline business loss size in row 4 by his/her estimate of a $30M business loss of releasing a Very Low quality upgrade, to generate a curve similar to the star curve in Figure 3 with a RELY investment sweet spot halfway between Nominal and High. Note that other analyses can be made to determine how much software quality investment is enough for other types of mission value loss reference points or alternative curves. We should point out that determining absolute business values such as $10M and $30M may not be easy, particularly if one has not done a business case for the project. However, even relative values can be used to obtain useful decision insights.

Furthermore, we can compare the results of value-based quality investment with value-neutral quality investment by their combined risk analyses. We present the results in Figure 4, using the high finance business case as an example. The decrease in $S_q(L)$ with testing time will be linear for the value-neutral testing, while the decrease in $S_q(L)$ 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. 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. Of course, the project will need to invest in some form of early requirements prioritization, such as business case analysis, stakeholder win-win negotiation, Total Quality Management, or agile methods story prioritization, but these generate other project advantages as well.
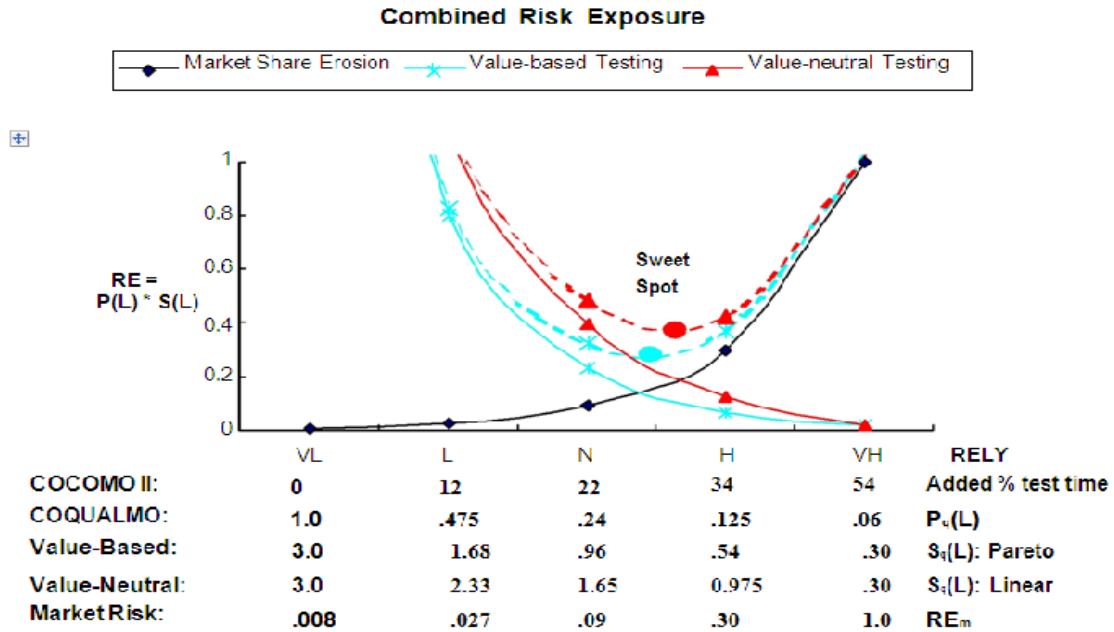
strate-

	VL	L	N	H	VH	RELY
COCOMO II:	0	12	22	34	54	Added % test time
COQUALMO:	1.0	.475	.24	.125	.06	$P_q(L)$
Value-Based:	3.0	1.68	.96	.54	.30	$S_q(L)$: Pareto
Value-Neutral:	3.0	2.33	1.65	0.975	.30	$S_q(L)$: Linear
Market Risk:	.008	.027	.09	.30	1.0	RE_m

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

6.2 Information Dependability Attribute Value Estimator (iDAVE)

We have extended the current iDAVE [3] model to support such risk analyses. It provides the default values of $S_q(L)$ and RE_m of each RELY rating for three business cases (i.e., early start-up, normal commercial and high finance). Users can also provide their own values for $S_q(L)$ and RE_m based on their project business case. After the user inputs the project size in SLOC and rates each COCOMO II cost driver except RELY according to their own project situation, iDAVE will generate the curve for combined risk exposure and help to locate the sweet spot for their software quality investment level. In addition, since the iDAVE tool is spreadsheet-based, it is easy to modify to handle other types of analyses for different situations discussed in [3] using different VERs, or to perform analyses of the sensitivity if the outcomes or sweet spots to unavoidable uncertainties in the input parameters or value functions.

4 CONCLUSION

In this paper, we present a value-based approach for project decision-makers to determine how much software quality investment is enough. This approach describes how the CO-COMO II and COQUALMO cost and quality estimation models and empirically-based business value estimating relationships (VERs) can be integrated to perform combined risk analyses in order to determine an approximate quantitatively-optimal software quality investment level and strategy for a project. (There may be qualitative factors such as key-stakeholder satisfaction that may make other levels or

strategies preferable). 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 iDAVE is also valuable for determining the relative payoff of value-based vs. value-neutral testing, which can be up to 40% higher for high-value applications, as shown in Figure 4.

Even with only approximate information on relative values, the models can provide a framework to help reason about quality investment tradeoffs and decisions. And for the future, some attractive research directions involve creating value-based counterparts for such value-neutral software quality technologies as test data generators, inspection checklists, defect closure metrics, and test plan aids.

REFERENCES

[1] B. Boehm, C. Abts, A.W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall, 2000. The COCOMO II software and documentation are available at : *http://sunset.usc.edu/research/COCOMOII/cocomo_main.html#downloads.*

[2] B. Steece, S. Chulani, and B. Boehm, "Determining Software Quality Using COQUALMO," in Case Studies in Reliability and Maintenance, W. Blischke and D. Murthy, Eds.: Wiley, 2002. The documentation of COQUALMO model is available at *http://sunset.usc.edu/research/coqualmo/coqualmo_main.html.*

[3] . 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. An initial spreadsheet tool of iDAVE is available at *http://sunset.usc.edu/cse/pub/research/iDAVE/iDAVE.zip*

[4] . Bullock, "Calculating the Value of Testing," Software Testing and Quality Engineering, May/June 2000, pp. 56-62.

[5] P. Gerrard, N. Thompson, Risk-Based E-Business Testing, Artech House, Inc., 2002

[6] R. Ramler, S. Biffl, and P. Gruenbacher, "Value-Based Management of Software Testing," in S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Gruenbacher, Value-Based Software Engineering, Springer Verlag, 2005

[7] P. Crosby, Quality is Free, McGraw Hill, 1979

[8] Inspired by "How much Software Quality Investment is Enough" by Li Guo Huang and Barry Boehm