Modelling the Quality Economics of Defect-Detection Techniques

Stefan Wagner
Institut für Informatik
Technische Universität München
Boltzmannstr. 3, D-85748 Garching b. München, Germany
wagnerst@in.tum.de

ABSTRACT

There are various ways to evaluate defect-detection techniques. However, for a comprehensive evaluation the only possibility is to reduce all influencing factors to costs. There are already some models and metrics for the cost of quality that can be used in that context. These models allow the structuring of the costs but do not show all influencing factors and their relationships. This paper proposes an analytical model for the economics of defect-detection techniques that can be used for analysis and optimisation of the usage of such techniques. In particular we analyse the sensitivity of the model and how the model can be applied in practice.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management; D.2.8 [**Software Engineering**]: Metrics; D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms

Economics, Verification, Reliability

Keywords

Software quality economics, quality model, defect-detection techniques, sensitivity analysis

1. INTRODUCTION

The quality of a software system can be described using different attributes such as reliability, maintainability etc. There are also various approaches to improve the quality of software with differing emphasis on these attributes. Constructive methods comprise one group that tries to improve the overall development process in order to prevent the introduction of faults. However, the prevalent approach is still to use analytical methods, also called defect-detection techniques, to find and remove faults. The main representatives of this approach are tests and reviews.

An often cited estimate [11] relates 50% of the overall development costs to testing. Jones [7] still assigns $30-40\%$ to quality assurance and defect removal. Hence, defect-detection techniques are a promising field for cost optimisations. However, to be able to optimise the usage of defect-detection techniques, we need an economical model first. There are some approaches that model software quality costs but mostly on a high level of abstraction. The effects of individual faults and the effectiveness of different defect-detection techniques regarding these faults are not taken into account. Also in [12] it is discussed that "Cost is clearly a central factor in any realistic comparison but it is hard to measure, data are not easy to obtain, and little has been done to deal with it." Rai et al. identify in [14] mathematical models of the economics of software quality assurance as an important research area. "A better understanding of the costs and benefits of SQA and improvements to existing quantitive models should be useful to decision-makers."

1.1 Problem

The underlying question is how we can optimally use defect-detection techniques to improve the quality of software. In particular, we investigate in this paper how the economical relationships of defect-detection techniques and quality can be modelled, which factors are the most important ones, and the applicability of such a model in practice.

1.2 Contribution

We propose an analytical model of the economics of defect-detection techniques incorporating different types of defect costs, the difficulty of finding a fault of different techniques and the probability of failure for a fault. This allows an evaluation of different techniques and gives a better understanding of the relationships. A sensitivity analysis of the model showed that the removal costs in the field and the failure probability have the strongest effect on the return on investment. From the technique-specific factors, the difficulties to find a specific defect are the most important ones. Furthermore, a model based on defect types is derived to allow a simpler application on real world projects to estimate and possibly predict the costs and benefits in a company or domain.

1.3 Outline

We start by describing software quality costs in general and our understanding of the various cost factors in Sec. 2.

©ACM, 2006. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proc. 4th Workshop on Software Quality (4-WoSQ), May 21, 2006, Shanghai, China
Copyright 2006 ACM 1-59593-085-X/06/0005 ...$5.00.

Sec. 3 proposes a model of the economics of defect-detection techniques that contains the costs and revenues. The application in practice is described in Sec. 4. Sec. 5 gives related work and final conclusions can be found in Sec. 6.

2. SOFTWARE QUALITY COSTS

Quality costs are the costs associated with preventing, finding, and correcting defective work. Based on experience from the manufacturing area quality cost models have been developed explicitly for software, e.g. [16]. These costs are divided into *conformance* and *nonconformance* costs, also called *control costs* and *failure of control costs*. The former comprises all costs that need to be spent to build the software in a way that it conforms to its quality requirements. This can be further broken down to *prevention* and *appraisal* costs. Prevention costs are for example developer training, tool costs, or quality audits, i. e. costs for means to prevent the injection of faults. The appraisal costs are caused by the usage of various types of tests and reviews.

The *nonconformance* costs come into play when the software does not conform to the quality requirements. These costs are divided into *internal failure* costs and *external failure* costs. The former contains costs caused by failures that occur during development, the latter describes costs that result from failures at the client. A graphical overview is given in Fig. 1. Because of the distinction between prevention, appraisal, and failure costs this is often called *PAF* model.

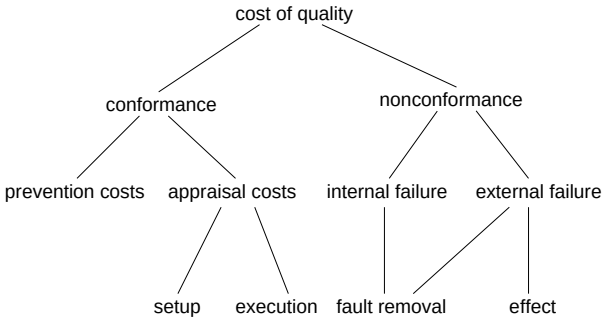

Figure 1: Overview over the costs related to quality

We add further detail to the PAF model by introducing the main types of concrete costs that are important for defect-detection techniques. Note that there are more types that could be included, for example, maintenance costs. However, we concentrate on a more reliability-oriented view. The appraisal costs are detailed to the *setup* and *execution* costs. The former constituting all initial costs for buying test tools, configuring the test environment, and so on. The latter means all the costs that are connected to actual test executions or review meetings, mainly personnel costs.

On the nonconformance side, we have *fault removal* costs that can be attributed to the internal failure costs as well as the external failure costs. This is because if we found a fault and want to remove it, it would always result in costs no matter whether caused in an internal or external failure. Actually, there does not have to be a failure at all. Considering code inspections, faults are found and removed that have never caused a failure during testing. It is also a good example that the removal costs can be quite different regarding different techniques. When a test identifies a failure, there needs to be considerable effort spent to find the corresponding fault. During an inspection, faults are found directly. Fault removal costs also contain the costs for necessary re-testing and re-inspections.

External failures also cause *effect* costs. These are all further costs with the failure apart from the removal costs. For example, *compensation* costs could be part of the effect costs, if the failure caused some kind of damage at the customer site. We might also include further costs such as loss of sales because of bad reputation in the effect costs but do not consider it explicitly because its out of scope of this paper.

3. ANALYTICAL MODEL

We propose a general, analytical model of defect-detection techniques. General is meant with respect to the various types of techniques. We mainly analyse different types of testing which essentially detect failures and static analysis techniques that reveal faults in the code or other documents. A model that incorporates all important factors for these differing techniques needs to use the universal unit of money, i. e. units such as euros or dollars. We divide the model in three main components:

- direct costs d_A
- future costs t_A
- revenues / saved costs r_A

The direct costs are characterised by containing only costs that can be directly measured during the execution of the technique. The future costs and revenues are both concerned with the (potential) costs in the field but can be distinguished because the future costs contain the costs that are really incurred whereas the revenues are comprised of saved costs.

In this section, we concentrate on an ideal model of quality economics in the sense that we do not consider the practical use of the model but want to mirror the actual relationships as faithfully as possible. We derive from this ideal model a practical version based on defect types in Sec. 4.1.

The main assumptions in the model are:

- Found faults are perfectly removed
- The amount or duration of a technique can be freely varied

The first assumption is often used in software reliability modelling to simplify the stochastic models. It states that each fault detected is instantly removed without introducing new faults. Although this is often not true in real defect removal, it is largely independent of the used defect-detection technique and the newly introduced faults can be handled like initial faults which introduces only a small blurring.

The second assumption is needed because we have a notion of time effort in the model to express for how long and with how many people a technique is used. This notion of time can be freely varied although for real defect-detection techniques this might not always make sense, especially when considering inspections or static analysis tools where a certain effort or none at all has to be spent.

We adapt the general notion of the difficulty of a technique A to find a specific fault i from [9] denoted by $\theta_A(i, t)$

as a basic quantity for our model and extend the difficulty function with the parameter t for the effort. The distribution of the difficulty w.r.t. the effort can then be defined depending on the fault and the technique. The geometric distribution can be used as a first approximation. In the following equations we are often interested in the case when the fault is detected at least once during the length of the technique application t_A. The probability that A detects i is $1 - \theta_A(i)$ and the probability that A never detects i is $\theta_A(i, t_A)$. Hence, we get the probability that i is at least detected once by $1 - \theta_A(i, t_A)$.

3.1 Direct Costs

The direct costs are those costs that can be directly measured from the application of a defect-detection technique. They are dependent on the length t of the application. With length we do not mean calendar time but effort measured in e. g. person days. Fig. 2 shows systematically the components of the direct costs.

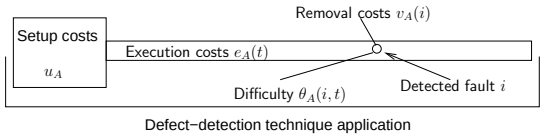

Figure 2: The components of the direct costs

Following [9] let

$$p_i = P(\text{fault } i \text{ is present in a randomly selected program}), \tag{1}$$

where i is the fault number. This is based on the idea that there is the set of all possible faults (labelled with the natural numbers) that might be in a program and the act of writing a specific program is a random selection. Then the faults have differing probabilities to be present in a program.

From this we can derive the following equation containing the three cost types.

$$d_A = u_A + e_A(t) + \sum_i p_i(1 - \theta_A(i,t))v_A(i), \tag{2}$$

where u_A are the setup costs, $e_A(t)$ the execution costs, and $v_A(i)$ the fault removal costs specific to that technique. Hence, we have for a technique its fixed setup costs, execution costs depending on the length of the technique and for each fault in the software removal costs if the technique is able to find it.

3.2 Future Costs

In case we were not able to find defects, these will result in costs in the future. We divide these costs into the two parts fault removal costs in the field $v_F(i)$ and failure effect costs $f_F(s)$. The latter contains all support and compensation costs as well as annoyed customers as far as possible.

$$t_A = \sum_i \pi_i \theta_A(i,t)(v_F(i) + f_F(s)), \tag{3}$$

where $\pi_i = P(\text{fault } i \text{ is activated by randomly selected input and is detected and fixed})$ [9]. Hence, it describes the probability that the defect leads to a failure in the field.

3.3 Revenues

We do not only have costs with defect-detection techniques but also revenues. These revenues are essentially saved future costs. With every fault that we find in-house we avoid higher costs in the future. Therefore, we have the same cost categories but look at the faults that we find instead of the ones we are not able to detect.

$$r_A = \sum_i \pi_i(1 - \theta_A(i,t))(v_F(i) + f_F(s)) \tag{4}$$

3.4 Combination

Typically, more than one technique is used to find defects. The intuition behind that is that they find (partly) different defects. These dependencies are often overlooked when the efficiency of defect-detection techniques is analysed. Nevertheless, this has a huge influence on the economics and efficiency. In our view, the notion of diversity of techniques from Littlewood et al. [9] is very useful in this context. The covariance of the difficulty functions of faults describes the similarity of the effectiveness regarding fault finding. An application of this model on quality economics can be found in [17]. We already use the difficulty functions in the present model and therefore are able to express the diversity implicitly.

For the direct costs it means that we sum up over all different applications of defect-detection techniques. We define that X is the ordered set of the applied defect-detection techniques. In each application we use Eq. 2 with the extension that we not only want the probability that the technique finds the fault but also that the ones before have not detected it.

$$d_X = \sum_{x \in X} \Bigg[u_x + e_x(t_x) + \sum_i \Big(p_i(1 - \theta_x(i,t_x)) \prod_{y<x} \theta_y(i,t_y) \Big) v_x(i) \Bigg] \tag{5}$$

The total future costs are simply the costs of each fault with the probability that it occurs and all techniques failed in detecting it.

$$t_X = \sum_i \Bigg[\pi_i \prod_{x \in X} \big(\theta_x(i,t_x)\big)(v_F(i) + f_F(s)) \Bigg] \tag{6}$$

The equation for the revenues uses again a sum over all technique applications. In this case we look at the faults that occur, that are detected by a technique and have not been detected by the earlier applied techniques.

$$r_X = \sum_{x \in X} \sum_i \Bigg[\Big(\pi_i(1 - \theta_x(i,t_x)) \prod_{y<x} \theta_y(i,t_y) \Big) \cdot \big(v_F(i) + f_F(s)\big) \Bigg] \tag{7}$$

3.5 ROI

One interesting metric based on these values is the return on investment (ROI) of the defect-detection techniques. If we look at the total ROI we have to use Eq. 5, Eq. 6, and Eq. 7 for the calculation.

$$\text{ROI} = \frac{r_X - d_X - t_X}{d_X + t_X} \tag{8}$$

This metric is suitable for a single post-evaluation of the quality assurance of a project. However, it cannot give an answer if the effort was cost-optimal. The comparability of this metric is limited because it strongly depends on the domain of the software.

3.6 Sensitivity Analysis

Any mathematical model should be subject to sensitivity analysis to assess the input factors and their effect on the output factors. In our case, we want to prioritise the input factors to (1) possibly simplify the model and remove or fix some of the input factors and (2) determine which of the factors should be measured with greatest accuracy. The latter is also an indicator to guide the future research in the sense that it pays off the most to analyse such factors.

There are various ways of sensitivity analysis depending on the characteristics of the model and the aims of the analysis. We use a *global* sensitivity analysis to rank the input factors with respect to the effect on the output factor. For this we use the Morris method as implemented in the SimLab tool. For details of the method and the tool see [15]. However, note that the method is computationally efficient in comparison to others but cannot quantify the difference between different factors but only give a ranking.

We use example and literature data of three testing techniques and a software with 24 defects as a basis. We cannot go into the details of the used distributions of the input factors but we mainly assumed triangular distributions with a mean value either from a case study or (if not available) from literature. The result from the Morris method is the mean μ and standard deviation σ of the sensitivity of each input factor. Our results are depicted in Fig. 3.

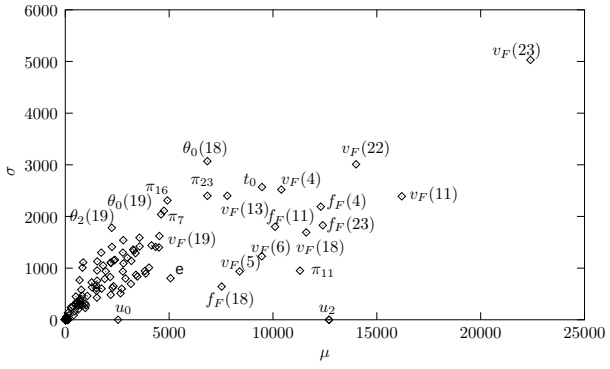

Figure 3: Sensitivity analysis using the Morris method

A detailed analysis is out of scope of this paper but we can easily notice that the main effect comes from the removal costs in the field (v_F) from various defects. Those factors lie mainly in the upper-right region. Other than the removal costs in the field, some of the effect costs (f_F) and failure probabilities (π) have stronger effects. From the difficulty functions (θ) mainly those of the first used technique have an influence. The setup costs are interesting because they have also a strong effect with nearly no standard deviation which means that they are constantly important. Surprisingly, only the spent effort (t) on the execution of the techniques is significantly influential from the first used technique. This suggests that the sequence of execution is important. This aspect needs further investigations.

Hence, we can conclude that a main emphasis should be put on the detailed measurement of fault removal costs in the field and the failure rate of the defects which are both independent from the used techniques. The main technique-dependent factors are the θ functions which should be analysed in more detail.

3.7 Discussion

The model so far is not suited for a practical application in a company as the quantities used are not easy to measure. Probably, we are unable to get values for θ of each fault and defect-detection technique. However, we can already use the model for interesting tasks besides sensitivity analysis. One application can be to analyse which techniques influence which parts of the model. For instance, in the automatic derivation of test cases from explicit behaviour models (model-based testing) is a relatively new technique for defect detection. This technique can be analysed and compared with traditional, hand-crafted test techniques based on our model. Two of the factors are obviously affected by model-based testing: (1) the setup costs are considerably higher than in hand-crafted tests because not only the normal test environment has to be set up but also a formal (and preferably executable) behaviour has to be developed. On the contrary, the execution cost per test case is then substantially smaller because the generation can be automised to some extent and the model can be used as an oracle to identify failures. Further influences on factors like the difficulty functions are not that obvious but need to be analysed. This example shows that the model can help to structure the comparison and analysis of defect-detection techniques.

4. PRACTICAL APPLICATION

As discussed in the preceeding section, the proposed model is difficult to use in practice. We derive a simpler model based on defect classes and describe possible applications.

4.1 Practical Model

For the simplification of the model, we use the following additional assumptions.

- Faults can be categorised in useful defect classes.
- Defect classes have specific distributions regarding their severity, removal difficulty, and failure probability.

We define $q_i(c)$ to be the probability that fault i belongs to the defect class c. In this way we do not have to look at individual faults but analyse and measure defect classes for which the determination of the quantities is considerably easier. We also estimate the number of faults contained in the software as $\bar{I}$ and ignore the p_i factor from the ideal model.

We give equivalents for all the equations from Sec. 3 in the following. We start with the direct costs of a defect-detection technique. Now we do not consider the ideal quantities but use average values for the cost factors. We denote this with a bar over the cost name.

$$d_A = \bar{u}_A + \bar{e}_A(t) + \sum_i \sum_c q_i(c))(1 - \theta_A(a,t))\bar{v}_A(c), \quad (9)$$

where $\bar{u}_A$ is the average setup cost for technique A, $\bar{e}_A(t)$ is the average execution cost for A with length t, and $\bar{v}_A(c)$ is the average removal cost in defect class c. Apart from using average values, the main difference is that we consider the probability of a fault to be in a specific fault class. The same applies to the revenues.

$$r_A = \sum_i \sum_c q_i(c)\pi_a(1 - \theta_A(a,t))(\bar{v}_F(c) + \bar{f}_F(c)), \quad (10)$$

where $\bar{f}_F(c)$ is the average effect costs of a fault of class c. Finally, the future costs can be formulated accordingly.

$$t_A = \sum_i \sum_c q_i(c)\pi_a\theta_A(a,t)(\bar{v}_F(c) + \bar{f}_F(c)) \quad (11)$$

The extension to the combined economics can be done accordingly.

4.2 Approaches

The practical model is suited to measure important aspects of defect-detection techniques in software engineering experiments by finding difficulty functions of certain techniques in certain domains. In software projects, the practical model can also help to optimise the future quality assurance by using the information from old projects.

4.2.1 In-House

The main idea is to predict the future economics based on the data from finished projects. The approach should then contain the following parts:

- Classify found faults
- Which technique found which fault?
- Which faults were found in the field?
- Estimate failure probability and costs for each fault

From this data, we can estimate the needed quantities of the model. This estimation process can have different forms. The failure probability can either be estimated by expert opinion or using field data. The cost data can be partly taken from effort measurements during development and from the field. Then we can try to answer two questions: What is the optimal length of a technique and what is the optimal combination? However, note that the results are in all cases dependent on the problem class and domain because they have a huge influence on the costs.

4.2.2 Domain-Specific

A second application could be to try to generalise the results of the model to a complete domain either from field studies or experiments. There are probably specific defect types in specific domains for which we might be able to collect data that is not only valid inside one company but for the whole domain. In this way, data from other companies could be used for optimisation purposes.

5. RELATED WORK

Our own previous work on the quality economics of defect-detection techniques forms the basis of this model. We formulated some simple relationships of cost factors and how this could be used in evaluating and comparing different techniques in [18]. This is refined in [19] and additional means to predict future costs are incorporated. Finally, the interplay between different techniques is added in [17] using some basic ideas of the model from Littlewood et al. [9].

The available related work can generally classified in two categories: (1) theoretical models of the effectiveness and efficiency of either test techniques or inspections and (2) economic-oriented, abstract models for quality assurance in general. The first type of models is able to incorporate interesting technical details but are typically restricted to a specific type of techniques and often economical considerations are not taken into account. The second type of models typically comes from more management-oriented researchers that consider economic constraints and are able to analyse different types of defect-detection but often deal with the technical details in a very abstract way. We cannot give a comprehensive overview of the related work but give some examples for both categories.

5.1 Efficiency Models

Pham describes in [13] various flavours of a software cost model for the purpose of deciding when to stop testing. It is a representative of models that are based on reliability models. The main problem with such models is that they are only able to analyse system testing and no other defect-detection techniques and the differences of different test techniques cannot be considered.

Kusumoto et al. propose in [8] a metric for cost effectiveness mainly aimed at software reviews. They introduce the concept of virtual software test costs that denote the testing cost that have been needed if no reviews were done. This implies that we always want a certain level of quality.

The economics of the inspection process are investigated in [1]. This work also uses defect classes and severity classes to determine the specific costs. However, it identifies only the smaller removal costs to be the benefit of an inspection.

An example of theoretical models of software testing is the work of Morasca and Serra-Capizzano [10]. They concentrate on the technical details such as the different failure rates. In this paper there is also a detailed review of similar models.

5.2 Economic Models

Several authors contributed views and aspects on the economics of software engineering and quality assurance. Erdogmus, for example, describes in [5] several cases which can be economically valued and some issues that must be considered.

In [16] a metric called *return on software quality (ROSQ)* is defined. It is intended to financially justify investments in quality improvement. The underpinnings of this metric are similar to the analytical model defined in this paper although there are significant differences. Firstly, it aims mainly on measuring the effects of process improvements, i. e. constructive quality assurance, whereas we concentrate on analytical quality assurance. Secondly, they base the calculations mainly on average defect content in the software and do not consider the important question if the faults lead

to failures.

Based on the general model for software cost estimation COCOMO, the COQUALMO model was specifically developed for quality costs in [4]. This model is different in that it uses only coarse-grained categories of defect-detection techniques. In our work, we want to analyse the differences between techniques in more detail.

Boehm et al. also present in [3] the iDAVE model that uses COCOMO II and COQUALMO. This model allows a thorough analysis of the ROI of dependability. The main difference is again the granularity. Only an average cost saving per defect is considered. We believe that analysing costs per defect type can improve estimates and predictions.

6. CONCLUSIONS

We propose an analytical model of quality economics with a strong focus on defect-detection techniques. This focus is necessary to be able to be more detailed than comparable approaches. In this way, we incorporate different cost types that are essential for evaluating defect-detection techniques and also a notion of reliability or the probability of failure. The latter is also very important because it is significant which faults are found in terms of reliability. This is obvious when considering that typically 80% of the downtime comes from 20% of the faults [2]. This distinguishes the model from more abstract approaches. On the other hand we have models derived from software reliability modelling. These models are typically simpler but can only be used on techniques where reliability models can be applied, i. e. mainly system tests. We aim to incorporate all types of defect-detection techniques.

The main weakness of our model is that the ideal model is not usable in real software projects. Hence, we derived the practical model that is based on defect classes. This gives us a greater data basis for each class. The problem here is, that it is not totally clear if this structuring in defect classes really is able to give useful distributions of the severity, removal difficulty, and failure probability. Furthermore, it strongly depends on how "good" these classes are defined and we currently have no requirements on the classes.

As future work, we consider working on support for the estimation of the needed quantities, especially the number of faults $\bar{I}$. The optimisation and sensitivity analysis must be worked on in more detail and effective tool support is essential to make the model applicable in practice.

7. ACKNOWLEDGMENTS

We are grateful to Sandro Morasca for detailed comments on the model and to Bev Littlewood for helping on the understanding of their diversity model. This research was supported by the *Deutsche Forschungsgemeinschaft (DFG)* within the project *InTime*.

8. REFERENCES

[1] S. Biffl. Hierarchical Economic Planning of the Inspection Process. In *Proc. Third International Workshop on Economics-Driven Software Engineering Research (EDSER-3)*. IEEE Computer Society Press, 2001.

[2] B. Boehm and V. R. Basili. Software Defect Reduction Top 10 List. *IEEE Computer*, 34(1):135–137, 2001.

[3] B. Boehm, L. Huang, A. Jain, and R. Madachy. The ROI of Software Dependability: The iDAVE Model. *IEEE Software*, 21(3):54–61, 2004.

[4] S. Chulani and B. Boehm. Modeling Software Defect Introduction and Removal: COQUALMO (COnstructive QUALity MOdel). Technical Report USC-CSE-99-510, University of Southern California, Center for Software Engineering, 1999.

[5] H. Erdopgmus. Valuation of Complex Options in Software Development. In *Proc. First Workshop on Economics-Driven Software Research (EDSER-1)*, 1999.

[6] G. J. Holzmann. Economics of Software Verification. In *Proc. 2001 ACM SIGPLAN–SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE'01)*, pages 80–89. ACM Press, 2001.

[7] C. Jones. *Applied Software Measurement: Assuring Productivity and Quality*. McGraw-Hill, 1991.

[8] S. Kusumoto, K. ichi Matasumoto, T. Kikuno, and K. Torii. A New Metric for Cost Effectiveness of Software Reviews. *IEICE Transactions on Information and Systems*, E75-D(5):674–680, 1992.

[9] B. Littlewood, P. T. Popov, L. Strigini, and N. Shryane. Modeling the Effects of Combining Diverse Software Fault Detection Techniques. *IEEE Transactions on Software Engineering*, 26(12):1157–1167, 2000.

[10] S. Morasca and S. Serra-Capizzano. On the Analytical Comparison of Testing Techniques. In *Proc. 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '04)*, pages 154–164. ACM Press, 2004.

[11] G. J. Myers. *The Art of Software Testing*. John Wiley & Sons, 1979.

[12] S. C. Ntafos. On Comparisons of Random, Partition, and Proportional Partition Testing. *IEEE Transactions on Software Engineering*, 27(10):949–960, 2001.

[13] H. Pham. *Software Reliability*. Springer, 2000.

[14] A. Rai, H. Song, and M. Troutt. Software Quality Assurance: An Analytical Survey and Research Prioritization. *Journal of Systems and Software*, 40:67–83, 1998.

[15] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice – A Guide to Assessing Scientific Models*. John Wiley & Sons, 2004.

[16] S. A. Slaughter, D. E. Harter, and M. S. Krishnan. Evaluating the Cost of Software Quality. *Communications of the ACM*, 41(8):67–73, 1998.

[17] S. Wagner. Software Quality Economics for Combining Defect-Detection Techniques. In *Proc. Net.Object Days 2005 (Node'05)*, pages 559–574. tranSIT GmbH, 2005.

[18] S. Wagner. Towards Software Quality Economics for Defect-Detection Techniques. In *Proc. 29th Annual IEEE/NASA Software Engineering Workshop (SEW-29)*, pages 265–274. IEEE Computer Society, 2005.

[19] S. Wagner and T. Seifert. Software Quality Economics for Defect-Detection Techniques Using Failure

Prediction. In *Proc. 3rd Workshop on Software Quality (3-WoSQ)*, pages 11–16. ACM Press, 2005.