A Literature Survey of 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

Over the last decades, a considerable amount of empirical knowledge about the efficiency of defect-detection techniques has been accumulated. Also a few surveys have summarised those studies with different focuses, usually for a specific type of technique. This work reviews the results of empirical studies and associates them with a model of software quality economics. This allows a better comparison of the different techniques and supports the application of the model in practice as several parameters can be approximated with typical average values. The main contributions are the provision of average values of several interesting quantities w.r.t. defect detection and the identification of areas that need further research because of the limited knowledge available.

Categories and Subject Descriptors

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

General Terms

Economics, Verification, Reliability

Keywords

Software quality economics, quality cost, cost/benefit, defect-detection techniques, literature survey

1. INTRODUCTION

The economics of software quality assurance (SQA) are a highly relevant topic in practice. Many estimates assign about half of the total development costs of software to SQA of which defect-detection techniques, i.e., analytical SQA, constitute the major part. Moreover, an understanding of the economics is essential for project management to answer the question how much quality assurance is enough. For example, Rai, Song, and Troutt [19] state that a better

understanding of the costs and benefits should be useful to decision-makers.

However, the relationships regarding those costs and benefits are often complicated and the data is difficult to obtain. Ntafos discusses in [18] that cost is a central factor but "it is hard to measure, data are not easy to obtain, and little has been done to deal with it". Nevertheless, there is a considerable amount of empirical studies regarding defect-detection techniques. The effectiveness and efficiency of testing and inspections has been investigated intensively over the last decades. Yet, we are not aware of a literature survey that summarises this empirical knowledge with respect to an economics model.

1.1 Problem

The main practical problem is how we can optimally use defect-detection techniques to improve the quality of software. Hence, the two main issues are (1) in which order and (2) with what effort the techniques should be used. This paper concentrates on the subproblem that the collection of all relevant data for a well-founded answer to these questions is not always possible.

1.2 Contribution

We review and summarise the empirical studies on various aspects of defect-detection techniques and software defects in general. The results of those studies are assigned to the different input factors of an economics model of analytical SQA. In particular, mean and median values of the input factors are derived to allow an easier application of the model in practice when not all factors are collectable. Furthermore, the found distributions can be used in further analyses of the model. Finally, the review reveals several areas that need further empirical research.

2. SOFTWARE QUALITY ECONOMICS

In this section, we introduce the general concept of quality costs for software. Based on that, we describe an analytical, stochastic model of the costs and benefits – the economics – of analytical SQA and finally possibilities of its practical application.

2.1 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. These costs are divided into *conformance* and *nonconformance* 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. The reason is that the removal of a detected defect always results in costs no matter whether it caused an internal or external failure.

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 it is out of scope of this paper.

2.2 An Analytical Model

We give a short overview of an analytical model of defect-detection techniques and refer to [23] for details. The model relates the discussed cost factors and other technical factors with the aim to analyse the economics of defect-detection techniques. In particular, it can be used to plan the quality assurance in a development project. Later we use the model as a basis for reviewing the empirical literature and hence describe only briefly the assumptions and equations.

2.2.1 General

We first describe 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 later simplify it for practical usages. The model is stochastic in the sense that it is based on expected values as basis for decision making.

We divide the model in three main components:

- Direct costs d_A
- Future costs o_A
- Revenues / saved costs r_A

The direct costs are characterised by containing only costs that can be directly measured during the application 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.

We adapt the general notion of the difficulty of an application of technique A to find a specific fault i from [15] denoted by $\theta_A(i)$ as a basic quantity for our model. In essence, it is the probability that A does not detect i. In the original definition this is independent of time or effort but describes a "single application". We extend this using the length of the technique application t_A. With length we do not mean calendar time but effort measured in staff-days, for example, that was spent for this technique application. Hence, the refined difficulty function is defined as $\theta_A(i, t_A)$ denoting the difficulty of A detecting i when applied with effort t_A.

Using this additional dimension we can also analyse different functional forms of the difficulty functions depending on the spent effort. This is similar to the informal curves shown by Boehm [3] describing the effectiveness of different defect-detection techniques depending on the spent costs. Actually, the equations given for the model above already contain that extended difficulty functions but they are not further elaborated. In [23] we considered several possible forms of the difficulty functions such as exponential or linear.

We also assume that in the difficulty functions the concept of defect classes is handled. A defect class is a group of defects based on the document type it is contained in. Hence, we have for each defect also its document class c, e.g., requirements defects or code defects. This has especially an effect considering that some techniques cannot be applied to all types of documents, e.g., functional testing cannot reveal a defect in a design document directly. It may however detect its successor in code.

This leads us to the further aspect that the defects occurring during development are not independent. There are various dependencies that could be considered but most importantly there is dependency in terms of propagation. Defects from earlier phases propagate to later phases and over process steps. We actually do not consider the phases to be the important factor here but the document types. In every development process there are different types of documents, or artifacts, that are created. Usually, those are requirements documents, design documents, code, and test specifications. Then one defect in one of these documents can lead to none, one, or more defects in later derived documents.

2.2.2 Components

We give an equation for each of the three components with respect to single defect-detection techniques first and later for a combination of techniques. Note that the main basis of the model are expected values, i.e., we combine cost data with probabilities.

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.

From this we can derive the following definition for the direct costs d_A:

$$d_A = u_A + e_A(t) + \sum_i (1 - \theta_A(i,t)) v_A(i), \qquad (1)$$

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.

If some defects are not found, these will result in costs in the future denoted by o_A. We divide these costs into the two parts fault removal costs in the field $v_F(i)$ and failure effect costs $f_F(i)$. The latter contain all support and compensation costs as well as annoyed customers as far as possible.

$$o_A = \sum_i \pi_i \theta_A(i,t)(v_F(i) + f_F(i)), \qquad (2)$$

where $\pi_i = P(\text{fault } i$ is activated by randomly selected input and is detected and fixed) [15]. Hence, it describes the probability that the defect leads to a failure in the field.

It is necessary to consider not only costs with defect-detection techniques but also revenues. These revenues are essentially saved future costs. With each 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. We denote the revenues with r_A.

$$r_A = \sum_i \pi_i (1 - \theta_A(i,t))(v_F(i) + f_F(i)) \qquad (3)$$

Because the revenues are saved future costs this equation looks similar to Eq. 2. The difference is only that we consider the faults that have not been found and hence use the probability of the negated difficulty, i.e., $1 - \theta_A(i,t)$.

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 ignored when the efficiency of defect-detection techniques is analysed. Nevertheless, they have a huge influence on the economics and efficiency. In our model this is expressed using the different difficulty functions for specific faults and techniques.

For the direct costs it means that we sum 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. 1 with the extension that we not only take the probability that the technique finds the fault into account but also that the ones before have not detected it. Here also the defect propagation needs to be considered, i.e., that not only the defect itself has not been detected but also its predecessors R_i.

For the sake of readability we introduce the abbreviation $\Theta(x, R_i)$ for the probability that a fault and its predecessors have not been found by previous – before x – applications of defect-detection techniques.

$$\Theta(x, R_i) = \prod_{y < x} \left[\theta_y(i, t_y) \prod_{j \in R_i} \theta_y(j, t_y) \right], \qquad (4)$$

hence, for each technique y that is applied before x we multiply the difficulty for the fault i and all its predecessors as described in the set R_i. The combined direct costs d_X of a sequence of defect-detection technique applications X is then defined as follows:

$$d_X = \sum_{x \in X} \left[u_x + e_x(t_x) + \sum_i \left((1 - \theta_x(i, t_x)) \Theta(x, R_i) \right) v_x(i) \right] \qquad (5)$$

The equation for the revenues r_X of several technique applications X 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 neither itself nor its predecessors have been detected by the earlier applied techniques.

$$r_X = \sum_{x \in X} \sum_i \left[\left(\pi_i (1 - \theta_x(i, t_x)) \Theta(x, R_i) \right) \left(v_F(i) + f_F(i) \right) \right] \qquad (6)$$

The total future costs are simply the costs of each fault with the probability that it occurs and all techniques failed in detecting it and its predecessors. In this case, the abbreviation $\Theta(x, R_i)$ for accounting of the effects of previous technique applications cannot be directly used because the outermost sum is over all the faults and hence the probability that a previous technique detected the fault is not relevant. The abbreviation $\Theta'(x, R_i)$ that describes only the product of the difficulties of detecting the predecessors of i is hinted in the following equation for the future cost o_X of several technique applications X.

$$o_X = \sum_i \left[\pi_i \prod_{x \in X} \theta_x(i, t_x) \underbrace{\prod_{y < x} \prod_{j \in R_i} \theta_y(j, t_y)}_{\Theta'(x, R_i)} (v_F(i) + f_F(i)) \right] \qquad (7)$$

2.2.3 ROI

One interesting metric based on these values is the *return on investment* (ROI) of the defect-detection techniques. The ROI – also called *rate of return* – is commonly defined as the gain divided by the used capital. Boehm et al. [4] use the equation $(\text{Benefits} - \text{Costs})/\text{Costs}$. To calculate the total ROI with our model we have to use Eqns. 5, 7, and 6.

$$\text{ROI} = \frac{r_X - d_X - o_X}{d_X + o_X} \qquad (8)$$

This metric can be used for two purposes: (1) an up-front evaluation of the quality assurance plan as the *expected* ROI of performing it and (2) a single post-evaluation of the quality assurance of a project. In the second case we can substitute the initial estimates with actually measured values. However, not all of the factors can be directly measured. Hence, also the post evaluation metric can be seen as an *estimated* ROI.

2.3 Practical Model

The ideal model can be used for theoretical analyses but is too detailed for a practical application. Hence, a simplified

version of this model is available that can be used to plan the quality assurance of a development project using historical project data. Details can be found in [23]. We only describe the additional assumptions and simplifications in the following.

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

- Faults can be categorised in useful defect types.

- Defect types have specific distributions regarding their detection difficulty, removal costs, and failure probability.

- The linear functional form of the difficulty approximates all other functional forms sufficiently.

We define τ_i to be the defect type of fault i. It is determined using the defect type distribution of older projects. In this way we do not have to look at individual faults but analyse and measure defect types for which the determination of the quantities is significantly easier.

In the practical model we assumed that the defects can be grouped in "useful" classes or defect types. For reformulating the equation it was sufficient to consider the affiliation of a defect to a type but for using the model in practice we need to further elaborate on the nature of defect types and how to measure them.

We also lose the concept of defect propagation as it was shown not to have a high priority in the analyses above but it introduces significant complexity to the model. Hence, the practical model can be simplified notably.

3. EMPIRICAL KNOWLEDGE

We review and summarise the empirical knowledge available for the quality economics of defect-detection techniques introducing the approach in general and then describing the relevant studies and results for each of the model factors for different types of techniques and defects in general. We can only give summaries here and refer to [22] for details.

3.1 General

The field of quality assurance and defect-detection techniques in particular has been subject to a number of empirical studies over the last decades. These studies were used to assess specific techniques or to validate certain laws and theories about defect-detection. Our focus lies on the economic relationships in the following.

3.1.1 Approach

This survey aims at reviewing and summarising the existing empirical work that can be used to approximate the input parameters of the economics model proposed in Sec. 2.2. For this we take all officially published sources into account, i.e. books, journal articles, and papers in workshop and conference proceedings. In total we reviewed 68 papers mainly following references from existing surveys and complementing those with newer publications. However, note that we only included studies with data relevant for the economics model. In particular, studies only with a comparison of techniques without detailed data for each were not taken into account.

In the literature review in the following sections, we structure the available work in three parts for dynamic testing,

review and inspection, and static analysis tools. We give a short characterisation for each category and describe briefly the available results for each relevant model input factor. We prefer to use and cite detailed results of single applications of techniques but also take mean values into account if necessary. We also summarise the combination of the results in terms of the lowest, highest, mean, and median value for each input factor.

We deliberately refrain from assigning weights to the various values we combine although some of them are from single experiments while others represent average values. The reason is that we often lack knowledge on the sample size used and either we would estimate it or ignore the whole study result. An estimate of the sample size would introduce additional blurring into the data and omitting data considering the limited amount of data available is not advisable. Hence, we assume each data set of having equal weight.

3.1.2 Difficulty

The difficulty function θ is hard to determine because it is complex to analyse the difficulty of finding each potential fault with different defect-detection techniques. Hence, we need to use the available empirical studies to get reasonable estimates. Firstly, we can use the numerous results for the effectiveness of different test techniques. The effectiveness is the ratio of found defects to total defects and hence in some sense the counterpart to the difficulty function. In the paper of Littlewood et al. [15], where the idea of the difficulty function originated, *effectiveness* is actually defined as

$$1 - E_{p^*}(\theta_A(i)), \qquad (9)$$

where E_{p^*} denotes a mean obtained with respect to the probability distribution p^*, i.e. the probability distribution of the presence of faults.

As a first, simple approximation we then define the following for the difficulty functions.

$$\theta_A = 1 - \text{effectiveness} \qquad (10)$$

The problem is that this is really a coarse-grained approximation that does not reflect the diversity of defect detection of different techniques. Hence, we also need to analyse studies that use different defect types in the sense of the practical model from Sec. 2.3.

3.2 Dynamic Testing

The first category of defect-detection techniques we look at is also the most important one in terms of practical usage. Dynamic testing is a technique that executes software with the aim to find failures.

3.2.1 Classification

There are various possibilities to classify different test techniques. One can identify at least two dimensions to structure the techniques. (1) The granularity of the test object and (2) the test case derivation technique. Fig. 2 shows these two dimensions and contains some concrete examples and how they can be placed according to these dimensions.

The types of test case derivation can be divided on the top level into (1) functional and (2) structural test techniques. The first only uses the specification to design tests, whereas the latter relies on the source code and the specification. In functional testing generally techniques such as equivalence

Figure 2: The two basic dimensions of test techniques

partitioning and boundary value analysis are used. Structural testing is often divided into control-flow and data-flow techniques. For the control-flow coverage metrics such as statement coverage or condition coverage are in use. The data-flow metrics measure the number and types of uses of variables.

On the granularity dimension we normally see the phases unit, module or component test, integration test, and system test. In unit tests only basic components of the system are tested using stubs to simulate the environment. During integration tests the components are combined and their interaction is analysed. Finally, in system testing the whole system is tested, often with some similarity to the later operational profile. This also corresponds to the development phases. Hence, the granularity dimension can also be seen as phase dimension.

3.2.2 Setup and Execution Costs

We look at the setup and execution costs in more detail in the following. For both cost types the empirical data is limited. However, this is not a great problem because this data can be easily collected in a software company during projects.

The setup costs are mainly the staff-hours needed for understanding the specification in general and setting up the test environment. For this we can use data from [12]. There the typical setup effort is given in relation to the size of the software measured in function points (fp). Unit tests need 0.50 h/fp, function tests 0.75 h/fp, system test 1.00 h/fp, and field tests 0.50 h/fp. We have no data for typical costs of tools and hardware but this can usually be found in accounting departments when using the economics model in practice.

In the case of execution costs its even easier than for setup costs as apart from the personnel costs all other costs can be neglected. One could include costs such as energy consumption but they are extremely small compared to the costs for the testers. Hence, we can reduce this to the typical, average costs for the staff. However, we also have average values per function point from [12]. There the average effort for unit tests is 0.25 h/fp, for function tests, system tests, and field tests 0.50 h/fp.

3.2.3 Difficulty

As discussed in Sec. 3.1.2, there are nearly no studies that present direct results for the difficulty function of defect-detection techniques. Hence, we analyse the effectiveness and efficiency results first. Those are dependent on the test case derivation technique used.

In the following we summarise a series of studies that have been published regarding the effectiveness of testing in general and specific testing techniques.

A summary of the found effectiveness of functional and structural test techniques can be found in Tab. 1. We can observe that the mean and median values are all quite close which suggests that there are no strong outliers. However, the range in general is rather large, especially when considering all test techniques. When comparing functional and structural testing, there is no significant difference visible.

Table 1: Summary of the effectiveness of test techniques (in percentages)

Type	Lowest	Mean	Median	Highest
Functional	33	53.26	48.85	88
Structural	17	54.78	56.85	89
All	7.2	49.85	47	89

The effectiveness gives a good first approximation of θ. The efficiency measures the number of detected defects per effort unit (staff-hours, for example). It cannot be used directly in the economics model but is also summarised as it is an important metric itself.

The found efficiencies of functional and structural test techniques are summarised in Tab. 2. We assume that one staff-hour consists of 60 staff-minutes. The results show that the data is quite homogenous because the means and medians are all equal or nearly equal and the ranges are rather small. Especially for functional testing it is only slightly above 1 defect/hour difference between the lowest and the highest value.

Table 2: Summary of the efficiency of test techniques (in defects per staff-hour)

Type	Lowest	Mean	Median	Highest
Functional	1.22	1.72	1.71	2.47
Structural	0.22	1.5	2.07	2.2
All	0.04	1.26	1.5	2.47

The first approximation of the difficulty functions is given in Tab. 3. We used the results of the effectiveness summary above. Hence, the observations are accordingly.

Table 3: First approximation of the difficulty functions for testing

Type	Lowest	Mean	Median	Highest
Functional	12	46.74	51.15	67
Structural	11	45.22	43.15	83
All	11	50.15	53	92.8

Finally, for a real application to our economics model we need to differentiate between different fault types. Basili and Selby analysed the effectiveness of functional and structural testing regarding different defect types in [2]. Tab. 4 shows the derived difficulties using the first approximation.

Table 4: Difficulties of functional and structural testing for detecting different defect types

	Functional Testing	Structural Testing	Overall
Initial.	25.0	53.8	38.5
Control	33.3	51.2	47.2
Data	71.7	73.2	74.7
Computat.	35.8	41.2	75.4
Interface	69.3	75.4	66.9
Cosmetic	91.7	92.3	89.2

It is obvious that there are differences of the two techniques for some defect types, in particular initialisation and control defects. As we are only aware of this single study it is difficult to generalise the results.

3.2.4 Removal Costs

The removal costs are dependent on the second dimension of testing (cf. Sec. 3.2.1): the phase in which it is used. This is in general a very common observation in defect removal that it is significantly more expensive to fix defects in later phases than in earlier ones. Specific for testing, in comparison with static techniques, is that defect removal not only involves the act of changing the code but before that of localising the fault in the code. This is simply a result of the fact that testing always observes failures for which the causing fault is not necessarily obvious. We cite the results of several studies regarding those costs in the following.

Some statistics of the data above on the removal costs are summarised in Tab. 5. We assume a staff-day to consist of 6 staff-hours and combined the functional and system test phases into the one phase "system test". The removal costs (or efforts) of the three phases can be given with reasonable results. A combination of all values for a general averages does not make sense as we get a huge range and a large difference between mean and median. This suggests a real difference in the removal costs over the different phases which is expected from standard software engineering literature.

Table 5: Summary of the removal costs of test techniques (in staff-hours per defect)

Type	Lowest	Mean	Median	Highest
Unit	1.5	3.46	2.5	6
Integration	3.06	5.42	4.55	9.5
System	2.82	8.37	6.2	20
All	0.2	8	4.95	52

3.3 Review and Inspection

The second category of defect-detection techniques under consideration are reviews and inspections, i.e. document reading with the aim to improve them.

3.3.1 Classification

We use the term *inspection* here in a broad sense for all kinds of document reading with the aim of defect-detection. In most cases *review* is used interchangeably. We can then identify differences mainly in the technical dimension, e.g., in the process of the inspections, for example whether explicit preparation is required. Other differences lie in the used reading techniques, e.g. checklists, in the required roles, or in the products that are inspected.

A prominent example is the formal or Fagan inspection that has a well-defined process with a separate preparation and meeting and defined roles. Another often used technique is the walkthrough. In this technique the moderator guides through the code but no preparation is required.

3.3.2 Setup and Execution Costs

The first question is whether reviews and inspections do have setup costs. We considered those costs to be fixed and independent of the time that the defect-detection technique is applied. In inspections we typically have a preparation and a meeting phase but both can be varied in length to detect more defects. Hence, they cannot be part of the setup costs. However, we have also an effort for the planning and the kick-off that is rather fixed. We consider those as the setup costs of inspections. One could also include costs for printing the documents but these costs can be neglected. Grady and van Slack describe in [9] the experience of Hewlett-Packard with inspections. They give typical time effort for the different inspection phases, for planning 2 staff-hours and for the kick-off 0.5 staff-hours.

The execution costs are for inspections and reviews only the personnel costs as long as there is no supporting software used. Hence, the execution costs are dependent on the factor t in our model. Nevertheless, there are some typical values for the execution costs of inspections.

We can derive some LOC-based statistics We assume for the sake of simplicity that all used varieties of the LOC metric are approximately equal. The results are summarised in Tab. 6. The mean and median values are all close. Only in code inspection meetings, there is a difference which can be explained by the small sample size. Note also that there is a significant difference between code and design inspections as the latter needs on average only half the execution costs. This might be explained by the fact that design documents are generally more abstract than code and hence easier to comprehend.

Table 6: Summary of the execution costs of inspection techniques (in staff-hours per KLOC)

Design	Lowest	Mean	Median	Highest
Preparation	3.6	4.68	4.68	5.76
Meeting	3.6	4.07	4.07	4.54
All	7.2	8.75	8.75	10.3
Code	Lowest	Mean	Median	Highest
Preparation	4.91	6.49	6.67	7.9
Meeting	3.32	7.02	4.4	13.33
All	6.67	13.2	11.15	22

Moreover, note that many authors give guidelines for the optimal inspection rate, i.e. how fast the inspectors read the documents. This seems to have an significant impact on the efficiency of the inspection. For example, in [8] the optimal bandwidth of the inspection rate is 1 ± 0.8 pages per hour where one page contains 300 words. As other authors give

similar figures, we can summarise this easily with saying that the optimal inspection rate lies about one page per hour. However, the effect of deviation from this optimum is not well understood. This, however, would increase the precision of models such as the one presented in Sec. 2.2.

3.3.3 Difficulty

Similar to the test techniques we start with analysing the effectiveness of inspections and reviews that is later used in the approximation of the difficulty.

We also summarise these results using the lowest, highest, mean, and median value in Tab. 7. We observe a quite stable mean value that is close to the median with about 30%. However, the range of values is huge. This suggests that an inspection is dependent on other factors to be effective.

Table 7: Summary of the effectiveness of inspection techniques (in percentage)

Lowest	Mean	Median	Highest
8.5	34.14	30	92.7

The efficiency relates the effectiveness with the spent effort. Again, this is not directly usable in the analytical model but nevertheless can give further insights into the relationships of factors.

The statistics for the efficiency of reviews and inspections can be found in Tab. 8. We do not distinguish different processes and reading techniques here because then we would not have enough information on these in most studies. The mean and median are close, therefore the data set is reasonable. We also observe a large range from 0.16 to 6 defects/staff-hour.

Table 8: Summary of the efficiency of inspection techniques (in defects per staff-hour)

Lowest	Mean	Median	Highest
0.16	1.87	1.18	6

Using the first, simple approximation, we can derive statistics for the difficulty of inspections in reviews in Tab. 9.

Table 9: Average difficulty of inspections (in percentages)

Lowest	Mean	Median	Highest
7.3	65.86	70	91.5

Analogous to the test techniques, we only have one study about effectiveness and defect types [2]. The derived difficulty functions are given in Tab. 10. Also for inspections large differences between the defect types are visible but a single study does not guarantee generalisability.

3.3.4 Removal Costs

The summary of the the removal costs can be found in Tab. 11. For the design reviews a strong difference between

Table 10: Difficulty of inspections to find different defect types

Initial.	35.4
Control	57.2
Data	79.3
Computat.	29.1
Interface	53.3
Cosmetic	83.3

the mean and median can be observed. However, in this case this is not because of outliers in the data but because of the small sample size of only four data points.

Table 11: Summary of the removal costs of inspections (in staff-hours per defect)

Phase	Lowest	Mean	Median	Highest
Requirements	0.05	1.06	1.1	2
Design	0.07	2.31	0.83	6.3
Coding	0.17	2.71	1.95	6.3
All	0.05	1.91	1.2	7.5

3.4 Static Analysis Tools

The third and final category is tool-based analysis of software code to automatise the detection of certain types of defects.

3.4.1 Classification

The term *static analysis tools* denotes a huge field of software tools that are able to find (potential) defects in software code without executing it. Those analysis tools use various techniques to identify critical code pieces. The most common one is to define typical bug patterns that are derived from experience and published common pitfalls in a certain programming language. Furthermore, coding guidelines and standards can be checked to allow a better readability. Also, more sophisticated analysis techniques based on the dataflow and controlflow are used. Finally, additional annotations in the code are introduced by some tools [7] to allow an extended static checking and a combination with model checking.

The results of such a tool are, however, not always real defects but can be seen as a warning that a piece of code is critical in some way. Hence, the analysis with respect to true and false positives is essential in the usage of bug finding tools.

There are only few studies about static analysis tools and hence we can only present limited empirical knowledge.

3.4.2 Setup and Execution Costs

There are no studies with data about the setup and execution costs of using static analysis tools. Still, we try to analyse those costs and their influence in the context of such tools.

The setup costs are typically quite small consisting only of (possible) tool costs — although there are several freely available tools — and effort for the installation of the tools

to have it ready for analysis.

The execution costs are small in the first step because we only need to select the source files to be checked and run the automatic analysis. For tools that rely on additional annotations the execution costs are considerably higher. The second step, to distinguish between true and false positives, is much more labour intensive than the first step. This requires possibly to read the code and analyse the interrelationships in the code which essentially constitutes a reviews of the code. Hence, the ratio of false positives is an important measure for the efficiency and execution costs of a tool.

In [25] we found that the average ratio of false positives over three tools for Java was 66% ranging from 31% up to 96%. In [11] a static analysis tools for C code is discussed. The authors state that sophisticated analysis of, for example, pointers leads to far less false positives than simple syntactical checks.

3.4.3 Difficulty

Static analysis techniques are evaluated in [10]. Interface consistency rules and anomaly analysis revealed 2 and 4 faults of 28, respectively. We also analysed the effectiveness of three Java bug finding tools in [25]. After eliminating the false positives, the tools were able to find 81% of the known defects over several projects. However, the defects had mainly a low severity. For the severest defects the effectiveness reduced to 22%, for the second severest defects even to 20%. For lower severities the effectiveness lies between 70% − 88%.

3.5 Defects

In this section we look at the quantities that are independent from a specific defect-detection technique and can be associated to defects. We are interested in typical defect type distributions, removal costs in the field, failure severities for the calculation of possible effect costs, and failure probabilities of faults.

3.5.1 Defect Introduction

The general probability that a specific possible fault is introduced into a specific program cannot be determined in general without replicated experiments. However, we can give some information when considering defect types. We can determine the defect type distribution for certain application types. Yet, there is only little data published. Sullivan and Chillarege described the defect type distribution of the database systems DB2 and IMS in [21]. Most of the defects were in assignment checking, data structures, and algorithm. Interface and timing defects constitute only a small share of the total number of defects.

Lutz and Mikulski used for defects in NASA software a slightly different classification of defects in [16] but they also have algorithms and assignments as types with a lot of occurrences. The most often defect type, however, is *procedures* meaning missing procedures or wrong call of procedures.

In [20] types and severities of software defects are described. We can observe that logical and data access defects account for most of the serious defects. Furthermore, most of the defects were defects in the specification.

As a summary, we can formulate that the defect types are strongly domain- and problem-specific and general conclusions are hard to make.

3.5.2 Removal Costs

In this section we analyse only the removal costs of defects in the field as during development we consider the removal costs to be dependent on the used defect-detection technique.

For the removal costs we have enough data to give reasonably some statistics in Tab. 12. Note that in this summary the mean and median are extremely different. The mean is more than twice the median. This indicates that there are outliers in the data set that distort the mean value. Hence, we look at a box plot of the data in Fig. 3.

Table 12: Summary of the removal costs of field defects (in staff-hours per defect)

Lowest	Mean	Median	Highest
3.9	57.42	27.6	250

Figure 3: Box plot of the removal costs of field defects in staff-hours per defect

The box plot in Fig. 3 shows two strong outliers that we can eliminate to get a more reasonable mean value. With the reduced data set we get a mean value of 27.24 staff-hours per defect and a median of 27 staff-hours per defect. Hence, we have a more balanced data set with a mean value that can be further used.

3.5.3 Effect Costs

The effect costs are probably the most difficult ones to obtain. One reason is that these are highly domain-specific. Another is that companies often do not publish such data as it could influence their reputation negatively. There is also one more inherent problem. It is often just not possible to to assign such costs to a single software fault. The highly complex configurations and the combination with hardware and possibly mechanics of software make such an assignment extremely difficult.

Yet, we cite two studies that published distribution of severity levels of defects. We consider the severity as the best available substitute of effect costs because more severe defects are probably more costly in that sense. However, this leaves us still with the need of a mapping of severity levels with typical effect costs.

Jones [12] states that the typical severity levels (1: System or program inoperable, 2: Major functions disabled or incorrect, 3: Minor functions disabled or incorrect, 4: Superficial error) have the following distribution:

1. 10% or 3%

2. 40% or 15%

3. 30% or 60%

4. 20% or 22%

In [6] is reported that six error types accounted for nearly 80% of the highest severity defects. Nine error types accounted for about 80% of the defects exposed by recovery procedures or exception handlers. They used ODC for classification.

3.5.4 Failure Probability

The failure probability of a fault is also one of the most difficult parts to determine in the economics model. Although there is the whole research field of software reliability engineering, there are only few studies that show representative distribution of such probabilities. The often cited paper from Adams [1] is one of the few exceptions. He mainly shows that the failure probabilities of the faults have an underlying geometric progression. This observation was also made in NASA studies reported in [17].

This relationship can also be supported by data from Siemens when used in a reliability model [24]. The geometric progression fitted reasonably on all analysed projects.

4. DISCUSSION

Some of the summaries allow a comparison over different techniques. Most interestingly, the difficulty of finding defects is different between tests and inspections with inspections having more difficulties. Tests tend on average to a difficulty of 0.45 whereas inspections have about 0.65. The static analysis tools are hard to compare because of the limited data but seem to be better in total but much worse considering severe defects.

The removal costs form a perfect series over the various techniques. As expected, the requirements reviews only need about 1 staff-hour of removal effort which rises over the other reviews to the unit tests with about 3.5 staff-hours. Over the testing phases we have again an increase to the system test with about 8 staff-hours. The field defects are then more than three times as expensive with 27 staff-hours. Hence, we can support the typical assumption that it gets more and more expensive to remove a defect over the development life-cycle.

We are aware that this survey can be criticised in many ways. One problem is clearly the combination of data from various sources without taking into account all the additional information. However, the aim of this survey is not to analyse specific techniques in detail and statistically test hypotheses but to determine some average values, some rules of thumb as approximations for the usage in an economics model. Furthermore, for many studies we do not have enough information for more sophisticated analyses.

Jones gives in [12] a rule of thumb: companies that have testing departments staffed by trained specialists will average about 10 to 15 percent higher in cumulative testing efficiency than companies which attempt testing by using their ordinary programming staff. Normal unit testing by programmers is seldom more than 25 percent efficient and most other forms of testing are usually less than 30 percent

efficient when carried out by untrained generalists. A series of well-planned tests by a professionally staffed testing group can exceed 35 percent per stage, and 80 percent in overall cumulative testing efficiency. Hence, the staff experience can be seen as one of the influential factors on the variations in our results.

5. RELATED WORK

The complete analytical model as described in this paper was published in [23].

The available related work to the economics model 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. Because of the limited space we do not cite those studies but refer to [23] for details.

There are also already some literature surveys on defect-detection techniques. Juristo et al. summarise in [13] the main experiments regarding testing techniques of the last 25 years. Their main focus is to classify the techniques and experiments and compare the techniques but not to collect and compare actual figures. In [5] several sources from the literature for inspection efficiency were used to build efficiency benchmarks. Laitenberger published a survey on inspection technologies in [14]. He also included data on effectiveness and effort but without relating it to a model.

6. CONCLUSIONS

We summarise the main results and contribution of the paper in the following and give directions for further research.

6.1 Summary

We reviewed and summarised the relevant empirical studies on defect-detection techniques that can be used to determine the input factors of an economics model of software quality assurance. The results of the studies were structured with respect to the technique they pertained and the corresponding input factor of the model. The difficulty function is the most complex factor to determine. We introduced two methods to obtain approximation of the factors for the three groups of techniques.

We observed that test techniques tend to be more efficient in defect detection having lesser difficulties but to have larger removal costs. A further analysis in the model might reveal which factor is more important. Furthermore, the removal costs increase also strongly considering different types of tests or reviews, i.e., during unit tests fault removal is considerably cheaper than during system tests. This suggests that unit-testing is very cost-efficient.

6.2 Further Research

We discussed an optimal inspection rate, i.e., the optimal effort per LOC regarding the efficiency of the inspection,

and noted that it is not well understood how a deviation from this optimal rate has effects on other factors in defect detection. Hence, further studies and experiments on this would be needed to refine the economics model and improve the analysis and prediction of the optimal quality assurance.

The difficulty of detecting different defect types with different detection techniques should be investigated more thoroughly. The empirical knowledge is extremely limited there although this would allow an improved combination of diverse techniques.

The effect costs are a difficult part of the failure costs. They are a highly delicate issue for most companies. Nevertheless, empirical knowledge is also important there to be able to estimate the influence on the total quality costs.

The collected empirical knowledge on the input factors can be used to refine the sensitivity analysis of the model that was done in [23]. A sensitivity analysis can be used to identify the most important input factors and their contribution to the variation in the output. The mean value and knowledge on the distribution (if available) can be used to generate more accurate input data to the analysis.

We will also extend the economics models by a size metric for better predictions because several factors, such as the execution costs, are dependent on the size of software.

7. REFERENCES

[1] E. N. Adams. Optimizing Preventive Service of Software Products. *IBM Journal of Research and Development*, 28(1):2–14, 1984.

[2] V. R. Basili and R. W. Selby. Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, SE-13(12):1278–1296, 1987.

[3] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

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

[5] L. Briand, K. E. Emam, O. Laitenberger, and T. Fussbroich. Using Simulation to Build Inspection Efficiency Benchmarks for Development Projects. In *Proc. 20th International Conference on Software Engineering (ICSE '98)*, pages 340–349. IEEE Computer Society, 1998.

[6] J. Christmansson and P. Santhanam. Error Injection Aimed at Fault Removal in Fault Tolerance Mechanisms – Criteria for Error Selection using Field Data on Software Faults. In *Proc. Seventh International Symposium on Software Reliability Engineering (ISSRE '96)*, pages 175–184. IEEE Computer Society, 1996.

[7] C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata. Extended Static Checking for Java. In *Proc. 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 234–245. ACM Press, 2002.

[8] T. Gilb and D. Graham. *Software Inspection*. Addison-Wesley, 1993.

[9] R. B. Grady and T. Van Slack. Key Lessons in Achieving Widespread Inspection Use. *IEEE Software*, 11(4):46–57, 1994.

[10] W. E. Howden. Theoretical and Empirical Studies of Program Testing. *IEEE Transactions on Software Engineering*, SE-4(4):293–298, 1978.

[11] R. Johnson and D. Wagner. Finding User/Kernel Pointer Bugs With Type Inference. In *Proc. 13th USENIX Security Symposium*, pages 119–134, 2004.

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

[13] N. Juristo, A. M. Moreno, and S. Vegas. Reviewing 25 Years of Testing Technique Experiments. *Empirical Software Engineering*, 9:7–44, 2004.

[14] O. Laitenberger. A Survey of Software Inspection Technologies. In *Handbook on Software Engineering and Knowledge Engineering*, volume 2, pages 517–555. World Scientific Publishing, 2002.

[15] 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.

[16] R. R. Lutz and I. C. Mikulski. Empirical Analysis of Safety-Critical Anomalies During Operations. *IEEE Transactions on Software Engineering*, 30(3):172–180, 2004.

[17] P. M. Nagel, F. W. Scholz, and J. A. Skrivan. Software Reliability: Additional Investigations into Modeling with Replicated Experiments. NASA Contractor Rep. 172378, NASA Langley Res. Center, Jun. 1984.

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

[19] 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.

[20] R. J. Rubey. Quantitative Aspects of Software Validation. In *Proc. International Conference on Reliable Software*, pages 246–251. ACM Press, 1975.

[21] M. Sullivan and R. Chillarege. A Comparison of Software Defects in Database Management Systems and Operating Systems. In *Proc. 22nd International Symposium on Fault-Tolerant Computing (FTCS-22)*, pages 475–484. IEEE Computer Society, 1992.

[22] S. Wagner. A Literature Survey of the Software Quality Economics of Defect-Detection Techniques. Technical Report TUM-I0614, Institut für Informatik, Technische Universität München, 2006.

[23] S. Wagner. A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques. In *Proc. International Symposium on Software Testing and Analysis (ISSTA '06)*, pages 73–83. ACM Press, 2006.

[24] S. Wagner and H. Fischer. A Software Reliability Model Based on a Geometric Sequence of Failure Rates. In *Proc. 11th International Conference on Reliable Software Technologies (Ada-Europe '06)*, volume 4006 of *LNCS*, pages 143–154. Springer, 2006.

[25] S. Wagner, J. Jürjens, C. Koller, and P. Trischberger. Comparing Bug Finding Tools with Reviews and Tests. In *Proc. 17th International Conference on Testing of Communicating Systems (TestCom'05)*, volume 3502 of *LNCS*, pages 40–55. Springer, 2005.