Towards Software Quality Economics for Defect-Detection Techniques*

Stefan Wagner
Institut für Informatik
Technische Universität München
Boltzmannstr. 3, D-85748 Garching, 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. The existing metrics for the effectiveness and efficiency of defect-detection techniques and experiences with them are combined with cost metrics to allow a more fine-grained estimation of costs and a comprehensive evaluation of defect-detection techniques. The current model is most suitable for directly comparing concrete applications of different techniques.

1. Introduction

The quality of a software system can be described based on several 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 to use analytical methods, also called defect-detection techniques, to find and remove faults.

An often cited estimate [24] relates 50% of the overall development costs to testing. Jones [9] still assigns 30 – 40% to quality assurance and defect removal. Because of this defect-detection techniques are a promising field for cost optimisations. One possibility is to concentrate the quality assurance on highly fault-prone components [31]. Another possibility that we describe in this paper is the evaluation of the efficiency of defect-detection techniques to optimise their usage. The ability to find faults has been studied by introducing effectiveness and efficiency metrics for defect-detection techniques. However, these metrics do not take the real costs into account, especially that different faults and failures lead to different costs. Software quality cost models on the other hand concentrate mainly on costs and make no use of the insights of the efficiency metrics.

For a closer analysis we first have to define the three most important terms in this context: *Failures* are a perceived deviation of the output values from the expected values whereas *faults* are the cause of failures in code or other documents. Both are also referred to as *defects*.

Problem. The problem we tackle in this paper is the difficulty of evaluating and comparing different defect-detection techniques. The ability to quantify the improvement caused by a specific technique is a prerequisite for optimisation.

Contribution. The contribution is an approach that structures and simplifies the determination of factors of quality economics especially for defect-detection techniques and thereby increases the precision of cost analyses. The structured approach can be used to evaluate different defect-detection techniques or the usage of defect-detection techniques in a specific environment.

Outline. A summary of efficiency metrics for defect-detection techniques is given in Sec. 2, software quality costs are explained in Sec. 3. The combination of these approaches is developed in Sec. 4. The combined approach is illustrated with an example in Sec. 5. Possibilities for a more general comparison are discussed in Sec. 6 and we conclude in Sec. 7. Related work is cited where appropriate.

2. Efficiency of defect-detection techniques

The effectiveness and efficiency of defect-detection techniques has been studied to a large extent [9, 11, 21]. *Effectiveness* is the level of improvement that a defect-detection technique obtains. The *efficiency* considers this level of

*This research was supported by the *Deutsche Forschungsgemeinschaft (DFG)* within the project *InTime* improvement in relation to the effort spent for the defect-detection technique. These metrics can be measured in various ways including counting the number of faults found, the achieved code coverage, or the fraction of the found faults in relation to all faults [30].

An approach especially considering the effect on reliability is proposed in [30] and compared with measures using pure fault counts. It shows that merely counting faults is not sufficient for evaluating defect-detection techniques. It furthermore suggests a combination of fault counts, fault MTTFs, and severities as a metric that combines reliability and safety aspects.

Especially the fault exposure ratio is considered to influence the effectiveness of testing [22]. It is a measure that describes the ratio of failures to faults, i.e. how many failures are caused by a fault. Sometimes this is given as an average for all faults but in this context we use the mean time to failure (MTTF) per fault that describes the caused failure rate of a single fault.

The severity of failures is already used as an effectiveness metric for inspections. A typical inspection data summary [6] distinguishes between minor and major defects. Also numerical metrics for severity are used, for example from 1 for a wrong text colour to 10 for endangering humans. Obviously a defect-detection technique is the more effective the more severe the defects are that it finds.

There are also different types of maintenance to be considered [29]: adaptive, corrective, preventive, and perfective. We concentrate on the effects and costs of occurred or prevented software failures hence only corrective maintenance effort is considered in the following. However, also the other types of maintenance are of importance and can have an influence on the quality costs. For example reading techniques often help to improve the architecture or readability of code and thereby reduce costs for adaption. Also bug finding tools mainly find defects that belong to the maintenance category [32].

3. Software quality costs

Quality costs are the costs associated with preventing, finding, and correcting defective work. Based on experience from the manufacturing area [10, 3] quality cost models have been developed explicitly for software [14, 28, 15]. 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 occurred 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.

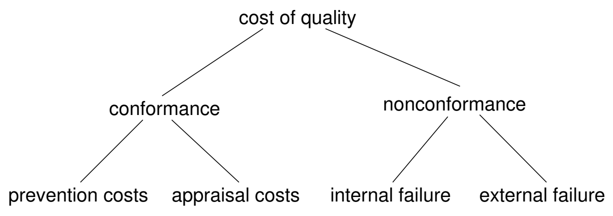

Figure 1. Overview over the costs related to quality

In [17, 28] further specific metrics are developed. The most important metric in this context is the *return on software quality (ROSQ)*. It is intended to financially justify investments in quality improvement. This investment, the *software quality investment (SQI)*, contains all the initial expenses for people, tools etc. The ongoing expenses are combined in the *software quality maintenance (SQM)*. The annual revenues resulting from the quality improvement derived from cost savings are called *software quality revenues (SQR)*. This is also different to other approaches such as [15] that do not incorporate revenues from cost savings.

The return of the software investment, i.e. the ROSQ can be determined by dividing the *net present value of the software quality cash flows (NPVCF)* by the *net present value of the initial investment and ongoing maintenance costs for the software quality initiative (NPVIC)*. The former includes SQI, SQR, and SQM, the latter only SQI and SQM. The calculation of the net present value requires a financial discounting factor r that reflects the company's weighted average cost of capital.

For a comparison of quality investments the *SQPI* is also interesting because it is the ratio of the present value of the difference between the software quality revenues and costs divided by the software quality investment. A value greater than 1 for the *SQPI* implies that the improvement creates value that exceeds its investment.

Critical in the model of [28] is that the basis for the cost determination are partly coarse-grained estimates of the defect density of the software. This metric does not contain any information about the MTTF and the severity of the defects. This should be more detailed to achieve higher precision with the estimates.

In [14] the model of software quality costs is set into relation to the Capability Maturity Model (CMM) [25]. The emphasis is hence on the prevention costs and how the improvement in terms of CMM levels helps in preventing failures.

Galin extends in [5, 4] the software quality costs with managerial aspects but the extensions are not relevant in the context of defect-detection techniques.

Guidelines for applying a quality cost model in a business environment in general are given in [13]. An empirical study of relating user satisfaction, cost, and quality can be found in [16]. Mandeville describes in [23] also software quality costs, a general methodology for cost collection, and how specific data from these costs can be used in communication with management.

Humphrey presents in [8] his understanding of software quality economics. The defined cost metrics do not represent monetary values but only fractions of the total development time. Furthermore, the effort for testing is classified as failure cost instead of appraisal cost.

Collofello and Woodfield propose in [2] a metric for the cost efficiency but do not incorporate fault MTTFs and severities explicitly. Furthermore, the metrics lacks consideration of the cost of capital and therefore the net present value is not used.

Pham describes in [26] 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 testing, not other defect-detection techniques and that the differences of different test techniques cannot be considered. However, they are highly useful as they allow a mathematical optimisation of the test time which is currently not possible with our model.

Based on the general model for software cost estimation COCOMO, the COQUALMO model was specifically developed for quality costs in [1]. This model is different in that it is aiming at estimating the costs beforehand and 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.

Holzmann describes in [7] his understanding of the economics of software verification. He describes some trends and hypotheses that are similar to ours and we can support most ideas although they need empirical justification at first.

Kusumoto et al. describe in [19, 18] 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.

4. Combination

This section develops a combination of the areas of software quality costs and efficiency metrics for defect-detection techniques to allow a detailed determination and comparison of the costs of specific techniques. We describe the concepts used in our extended cost model, propose a new economics metric, and present the procedure for determining the metric.

4.1. Concepts

We develop a quality economics model especially tailored for defect-detection techniques but before we go into mathematical details, the concepts relevant for the model have to be identified.

Cost categories. For the combination of the efficiency and cost metrics tailored for defect-detection techniques, we go through the cost categories and identify which parts of the efficiency metrics correspond to it and are able to help in estimating the cost. The *prevention costs* are out of scope for this combination because defect-detection techniques typically find faults and do not prevent them. Design or architecture reviews that can identify defects before they are introduced into code could be seen as an exception because they prevent faults from being introduced into code. This depends on the point of view concerning faults. We will classify defects in the design also as faults because they also have a corresponding change effort. Therefore, we attribute costs for design reviews to *appraisal costs* in the following.

Furthermore, all costs that are directly related to testing and inspecting the software can be assigned to the *appraisal costs*. These are, for example, the purchase of a test tool, personnel costs for the preparation of the test environment, generation, execution, and analysis of test cases etc. The costs for the debugging process can be attributed to the *internal failure* costs. For this we introduce the *change effort* for a fault. It includes the personnel costs for finding and removing it but also costs for debugging tools, re-inspection, and re-testing. This shows that the change effort changes over time. The later the fault is detected and removed, the more costs it causes. For example a design fault is easily changed during design but during testing it requires a change of the design documents, code, and test cases as well as re-inspection and re-testing. See Fig. 2 for a typical change of the nonconformance costs including the change effort over the life cycle of a software.

The main contribution of the efficiency metrics lies in the *external failure* costs. This includes also the change effort that can be significantly higher than for internal failures costs because the removal of defects from software that is

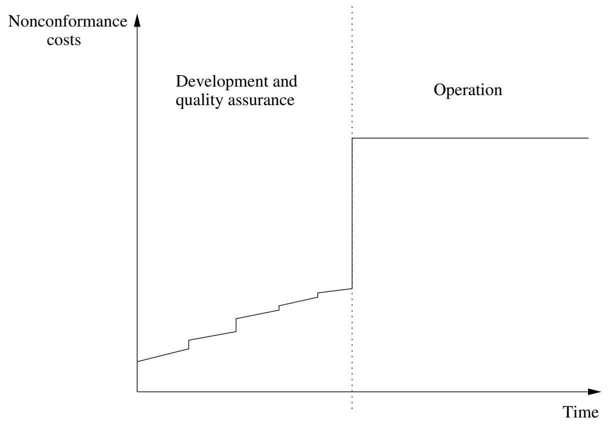

Figure 2. Typical change of the nonconformance costs over the development project

used in the field can include expensive deployment. Furthermore compensation for damages on behalf of the client and lost sales because of the failure must be taken into account. Especially for these costs the fault MTTF and severity are interesting measures. They can help in estimating (1) how often failures will occur and (2) how severe the consequences will be. Depending on the detailedness of these single estimates the cost can be estimated in more detail.

Period. A further concept we need to introduce is that of a *period*. This means a specific time interval in the development process that contains a defect-detection technique. It does not have to correspond to a classical waterfall phase and can be used with an iterative process as well. The point is only to be able to (1) estimate the change effort depending on the point in time and (2) have a basis for the calculation of the present net values. The first is important because the additional work increases the later the fault is removed. For example if a design fault is detected early only the design documents need to be changed, later also the code and test cases have to be adapted. The latter is important because to be able to compare costs the cash flows need to be analysed in dependence of the point in time when they occur.

Interrelations. All the associations are described in a class diagram in Figure 3. A defect-detection technique (DDT) is performed in a specific period of the development process. For each DDT there are internal failures that it detected and external failures that passed the defect-detection technique. These failures have their cause in faults with a measured or estimated fault MTTF and an estimated severity. A fault has an associated change effort depending on the period in which it was found.

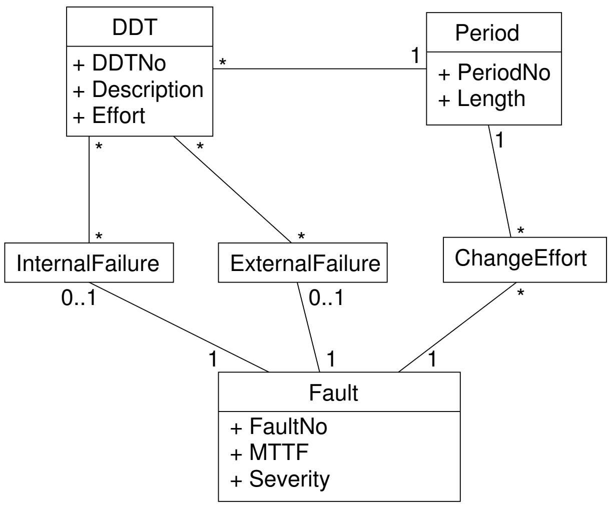

Figure 3. Class diagram showing the associations between the concepts

4.2. Cost metrics

The basis for the analysis of the quality costs is the net present value of the cash flows (NPVCF), i.e. the revenues and costs during the life cycle of the software that are related to quality. This metric is sufficient to judge the earning power and also to compare different defect-detection techniques. For further analyses other metrics from [28] such as ROI or SQPI can be used.

In contrast to [28] we analyse NPVCF in greater detail. For a specific defect-detection technique we have fixed initial investments called *fixed appraisal costs (FAC)* that contain for example tool or workstation costs. This is called SQI in [28] but we want to change the name to have a closer relation to the PAF model. The software quality maintenance (SQM) from [28] is the dynamic part of the appraisal costs *(DAC)*, e.g. personnel for tests and inspections.

The software quality revenues (SQR) are split into three distinct parts following the concepts developed above. Firstly, the change effort *(CE)* has to be considered. As mentioned above it is differing depending on the amount of document changes, re-testing and re-inspection that has to be done. This amount depends on the period in which the fault is detected. The name is not oriented on the PAF model because it is part of the internal as well as the external failure costs.

Secondly, the further costs *(FC)* of external failures such as compensation costs and lost sales are part of SQR. This is again not named after the PAF model as it only constitutes part of the external failure costs. The other are some of the CE costs. Finally, on the positive side are the costs that have been saved by the defect-detection technique. The *saved* *costs (SC)* are the costs that would have been caused by the faults that were found, i.e. their CE and FC.

The FAC are supposed to be made at present and have therefore already the present value. All the other costs and revenues have to be discounted to the present value. For this we need a discount factor D, called r in [28], that represents an average cost of capital. The calculation can easily be based on the periods defined above for the change effort for internal failures because we know the time that has passed until a defect was found and repaired. Therefore all dynamic costs, i.e. not the FAC, are annotated with the period they occur in.

The more difficult part is to estimate the time until an external failure will occur and will result in further costs. This can be estimated based on the fault MTTF estimated earlier. Similarly the revenues, i.e. the saved costs, must be handled accordingly. All this assumes that we have a fixed granularity for the periods that is used also for the estimate of time periods for external failures. Based on this the following equation can be used to calculate the net present value of the cash flows.

$$NPVCF = -FAC + \sum_{i=1}^{P} \frac{SC_i - CE_i - FC_i - DAC_i}{(1+D)^i}, \quad (1)$$

where SC_i are the saved costs in period i, CE_i the change effort in period i, FC_i the further costs of external failures in period i, D the discount factor relevant for the environment, and P the number of periods in the whole life cycle.

For the determination of SC_i the same basic values can be used. The following equation describes the calculation based on the idea that the revenues, i.e. the saved costs, are the costs that the removed faults would have caused.

$$SC_k = \sum_{j=k+1}^{P} CE_j + FC_j, \quad (2)$$

where CE_j is the change effort and FC_j the further costs of the defects that were found by the defect-detection technique and would have occurred in period j.

Threats to validity. Together with the proposal of a metric one has to investigate the potential threats to validity. The main problem is to get accurate values either from accounting or from estimates. The estimates can be supported by known or estimated efficiency metrics but still there is a great potential for errors. Another problem is the sequential usage of technique. If two or more techniques are analysed that are not completely independent and used one after the other, they influence each other. The faults found and removed because of the earlier technique cannot be found from the later. However, the uncertainty of the expert opinion can be accounted for by using, for example, Monte-Carlo simulation.

4.3. Procedure

The procedure for using the economics model to compare defect-detection techniques starts with the compilation of a list of all the faults that were found by the defect-detection techniques. They are simply numbered and enriched with additional information. This information should at least be estimations of the MTTF, the severity, and the change effort. The latter is divided into the specific values for each period. Having this list, the faults can be assigned to the set of internal and external failures for each technique. This can be used to calculate the costs regarding internal and external failures for each technique.

The appraisal costs are also needed to get the complete cost calculation. We therefore calculate the appraisal costs by adding at least tool and personnel costs. Some of this information is available from defect databases and accounting. The remaining values have to be estimated by experts. Especially for the revenues estimating is the only possibility. We do not have hard data about the cost savings because the failures do not occur. However, they cannot be ignored because they are the main benefits of using defect-detection techniques. Therefore estimates are important and necessary for the quality economics of DDTs. The comparison can finally be based on the costs only or on further metrics such as the *ROSQ* or *SQPI* (cf. Sec. 3). An example can be found in the following Sec. 5.

Sequential usage of techniques. If defect-detection techniques are analysed that are used one after the other, the procedure has to be slightly changed. The defects that were revealed and removed before a technique is used cannot be counted as external failures for that technique. Only faults found after the technique under investigation are external failures.

As mentioned above this blurs the data for all techniques because if an extremely effective technique was used that found a lot of defects that were removed before the next technique is applied, the next one cannot be as effective as it normally would be with all faults still in the software. However, this is an inherent problem. What we can do is to experiment with different orders of the techniques and with different amounts of effort spent for each technique. This way we can find the optimal combination of techniques.

5. Example

The example shows an usage of the above developed combination of efficiency metrics for defect-detection techniques and cost of software quality metrics. We loosely base this example on a project with a large automotive manufacturer but the data has been changed to some extent because of confidentiality reasons.

We want to compare three different defect-detection techniques that are used for system testing of an embedded system. The techniques are independently applied to the same version of the system. All techniques together are able to find 24 faults. These are listed in Tab. 1 where the faults are numbered and are accompanied by estimates of the MTTF in *high*, *average*, and *low*, the severity in *critical* and *noncritical*, and the change effort in person hours. We only include two values for the change effort because all three techniques were applied independently in the same period. Therefore, we have one value for the test period and one for the operation period.

These faults now have to be assigned to the defect-detection techniques. We assign all the faults that were found by a technique to its internal failure set and the ones that were not found to its external failure set. This is the result of the assumption that all faults that were not found by a specific technique lead to failures in the field with the probability determined by the MTTF. In case there is additional information about real field failures, it could be incorporated. The mapping for the three defect-detection techniques T_1, T_2 and T_3 is also described in Tab. 1. The letters *i* and *e* denote that a fault belongs to the internal or external failure set, respectively.

The mapping is used to calculate the costs for internal and external failures for each defect-detection technique. It is important because for an internal failure the costs are only the change effort, whereas external failures also result in compensation costs and image loss. In some domains the change effort can even be different depending on this distinction. If the deployment is costly as in many embedded systems the cost for updates can be significant.

The concrete figures for the costs have to be estimated by an expert. However, the earlier compiled information for each fault can guide these estimations. Critical faults with a high MTTF will have higher costs than noncritical faults with a low MTTF. Depending on the detailedness of the MTTF and severity this can even be expressed numerically. For the sake of the example we choose 1000 monetary units as basic costs of an external failure and multiply it by 0.5 for MTTF *low*, 1 for *average*, and 2 for *high*. The severity *critical* increases the costs with the factor 10. Furthermore we assume the cost of a person hour for the removal of a fault as 100 monetary units. Finally, we assume that the techniques have different requirements on the used tools, therefore having different fixed and dynamic appraisal costs.

For the calculation of the net present values, we use a discounting factor D of 5% and assume a granularity of the periods of 1 year. Therefore the tests that are compared are all in period 1. For the calculation of the external failure costs we need to estimate when the faults would result in failures. We use the fault MTTF again for this estimation. Faults with a fault MTTF of *high* are assumed to lead to a failure in period 2, of *average* in period 4, and of *low* in period 8.

We do not go into details of the calculation of all the values but use one example to illustrate the approach. We look at period 4 of technique T_1 in more detail. In this period the faults 1, 6, 12, 14, 20, and 23 could lead to a failure if it has not been found. In the case of T_1 only fault number 1 leads actually to a failure because the others have been detected in period 1. Therefore, we have a negative cash flow of the change effort and the further failure costs only for fault 1. This consists of 3800 for the change effort and 10000 for further compensations. However, we saved costs for the other five faults that do not lead to a failure. Therefore, we add up the change effort and further failure costs for all of these. This results in 41400. The 13800 have to be subtracted from this and the result have to be discounted with the factor 5%. The final result is then 22706.59. This is the net present value of the cash flows of period 4.

All calculated cash flows are summarised in Tab. 2. We included the values for the fixed and dynamic appraisal costs (FAC and DAC_1), the discounted cash flows for each period where failures could occur and finally the total net present value of the cash flows (NPVCF).

These calculations show directly that the technique T_2 returns the most value overall although it has the highest appraisal costs. However, it finds the most faults and therefore the external failure costs decrease significantly.

6. General comparison

The economics model developed in the previous sections is suited to measure and compare different concrete applications of defect-detection techniques economically. This is a reasonable first step but the underlying motivation is to compare defect-detection techniques in general and find characteristics that allow the optimal usage and combination of such techniques.

Most approaches described in Sec.2 and Sec. 3 reduce the evaluation of a defect-detection technique to a single metric. In our view the evaluation of a technique should be at least two-dimensional. We should not only evaluate the net present value of the cash flows but also the change of these cash flows when spending different amounts of efforts for that technique.

We call this the characteristic curve of the cash flows in relation to the effort spent for the technique. Following the intuition, each technique starts with negative cash flows as we have initial investments. With further usage of the tech-

Number	MTTF	Severity	Change eff.		Mapping		
			Test	Oper.	T_1	T_2	T_3
1	average	critical	8	38	e	i	e
2	low	noncritical	3	33	i	i	i
3	low	critical	16	46	i	i	i
4	low	noncritical	3	33	i	i	i
5	high	critical	20	50	i	i	e
6	average	noncritical	3	33	i	i	i
7	high	noncritical	9	39	i	i	i
8	low	critical	1	31	i	i	i
9	low	critical	12	42	i	i	i
10	low	critical	4	34	e	i	e
11	low	critical	7	37	e	i	i
12	average	critical	4	34	i	i	i
13	low	noncritical	5	35	i	i	e
14	average	noncritical	8	38	i	i	e
15	low	critical	19	49	i	i	e
16	low	critical	2	32	i	e	e
17	low	critical	8	38	i	i	e
18	low	critical	10	40	e	e	e
19	high	critical	5	35	i	i	e
20	average	critical	8	38	i	i	e
21	low	uncritical	12	42	e	i	e
22	low	critical	4	34	e	e	e
23	average	uncritical	11	41	i	i	i
24	high	critical	8	38	i	i	i

Table 1. Summary of all the found faults and their mapping to the defect-detection techniques

Tech.	FAC	DAC_1	Period 1	Period 2	Period 4	Period 8	NPVCF
T_1	15,000	18,000	-31,642.86	70,929.71	22,706.59	17,733.19	64,726.63
T_2	15,000	20,000	-36,447.62	70,929.71	45,413.18	36,143.22	101,038.48
T_3	1,000	12,000	-19,128.57	-17,052.15	-7,897.94	-12,318.48	- 57,397.15

Table 2. The estimated cash flows for each defect-detection technique

nique the cash flows become positive rapidly until it reaches an area of satisfaction where less and less faults are found. Finally, with still more effort only the costs rise but no revenues are generated because only few or no new faults are found. Hence, the sum of the cash flows decreases. An example curve following this intuition is depicted in Fig. 4.

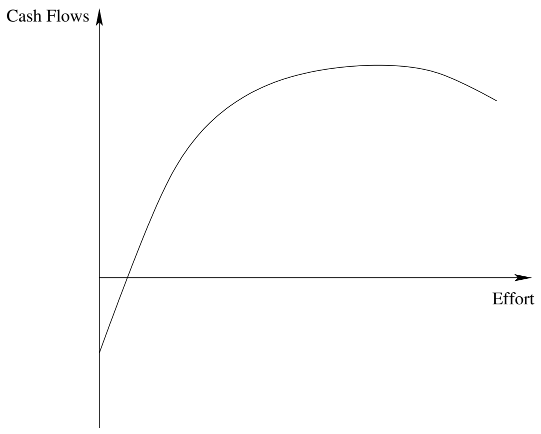

Figure 4. Typical characteristic cash flows for a technique

The shape of this curve is also justified by the so-called S-curve of software testing [12, 7] that shows that the search for defects becomes less effective in the end with more amount of time spent. We believe this curve is valid in principle for all defect-detection techniques although some specific techniques such as static analysis tools have extreme shapes in this family of curves.

The establishment of such characteristic curves for several techniques requires extensive measurement experiments. It implies the usage of each technique in different contexts and with differing efforts spent. However, having such general curves allows the early planning of quality assurance using optimisation techniques. The different characteristics can be combined to form the curve with the maximum of possible positive cash flows.

For this it is necessary to normalise different curves from different projects and techniques to be able to compare different curves. This can be done by dividing each axis by a size metric of the software like lines of code (LOC) or function points. Another possibility to make the curves more comparable is to use the internal rate of return instead of the net present value of the cash flows. This quantifies the earning power in relation to the investments which then also would account for the problem of safety-critical systems. Those systems yield significantly higher saved costs than other systems because a safety-critical failure would lead to enormous costs. It is not possible to use this for the effort axis. An analogous approach and an alternative to the size metrics is to set the effort for the defect-detection technique into relation to the total effort spent for the software development.

We also need to investigate on which factors the curves depend. It might be the case that the curves are different in different domains or depending on the experience of the test engineers. Further influencing factors might be the programming language, specific tool support, difficulty of the specific programming problems, algorithms, communication and so on.

Another problem is that the curve will change if other techniques are used before the technique under consideration. The difference is mainly that the curve is pushed down depending on the diversity of all the applied techniques. Earlier used techniques find faults that the technique under consideration would also be able to find. Therefore, the level of positive cash flows cannot be reached. We are currently investigating the usage of the model of diversity of defect-detection techniques in [20] to account for these influences.

7. Conclusions

We present an approach for comparing defect-detection techniques that is based on costs but takes information from efficiency metrics into account. This is a first step towards comprehensive quality economics for defect-detection techniques. The issues for constructive techniques, i.e. fault prevention, are substantially different. Therefore, we concentrated on analytical techniques, i.e. defect-detection techniques in this paper.

We reviewed the current knowledge on cost models for software quality and detailed and extended existing cost models using experience from efficiency metrics for defect-detection techniques. In detail the severity and fault MTTF of each fault is used to estimate the costs as well as the revenues for the techniques. All this information is set into relation to the development periods to be able to calculate the net present values and thereby determine the economic value of the technique.

The resulting quality cost model can be used for two purposes: (1) compare different defect-detection techniques in an experimental setting or (2) the usage of techniques in a given environment. The latter needs experimenting with differing orders and efforts for the techniques to be able to find an optimal resource usage.

Future Work. Based on the evaluation of model-based testing in [27] a further study is planed that emphasises the cost aspects of the technique. In this study the developed quality cost model will be used for the evaluation.

For all of this to yield solid information, we need to assess the impact of model uncertainties. This is important as the whole revenues are based only on expert opinion which are unreliable. Sensitivity analysis or Monte Carlo simulation are approaches we currently investigate for this.

Another area we have to investigate further is the fraction of faults that are not known at the time of the analysis using our cost model. Typically we do not want to wait several years to be confident that most faults have been found and to be able to analyse the used defect-detection techniques. Therefore, it would be helpful to have support on estimating the remaining faults in the software to incorporate their influence on the costs. A possible solution to this is to use software reliability models as in [33].

Furthermore, the costs for wrongly identified faults, i.e. faults that are not really faults, have to be considered. A significant part of defects reported are actually not defects but have other reasons. This nevertheless results in costs in analysing the defect that have to be considered in the model.

A main point for further research are the incorporation of all types of maintenance costs into the cost model. Especially for software the maintenance costs are important because software is in general changed easily. Therefore adaptive, preventive, and perfective maintenance need their place in the cost model as well. This means that these activities would be added to the cost side but also the resulting revenues when corresponding changes are simpler have to be included.

Also the opportunity costs should be included in the model. These are costs that result from failures that cannot be easily measured such as lost sales or image loss. Means to quantify and predict opportunity costs are difficult to find but we could also use expert opinion in combination with Monte Carlo simulation.

We also want to use the theoretical model of combining diverse techniques from [20] to incorporate those effects in our cost model. Having such a mechanism in our model would allow us to optimise the combination of techniques based on economic judgment.

References

[1] 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.

[2] J. Collofello and S. Woodfield. Evaluating the Effectiveness of Reliability-Assurance Techniques. *Journal of Systems and Software*, 9(3):191–195, 1989.

[3] A. Feigenbaum. *Total Quality Control*. McGraw-Hill, 3rd edition, 1991.

[4] D. Galin. *Software Quality Assurance*. Pearson, 2004.

[5] D. Galin. Toward an Inclusive Model for the Costs of Software Quality. *Software Quality Professional*, 6(4):25–31, 2004.

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

[7] G. 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.

[8] W. Humphrey. *A Discipline for Software Engineering*. The SEI Series in Software Engineering. Addison-Wesley, 1995.

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

[10] J. Juran, editor. *Quality Control Handbook*. McGraw-Hill, 4th edition, 1988.

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

[12] S. Kan, J. Parrish, and D. Manlove. In-Process Metrics for Software Testing. *IBM Systems Journal*, 40(1):220–241, 2001.

[13] C. Kaner. Quality Cost Analysis: Benefits and Risks. *Software QA*, 3(1), 1996.

[14] S. Knox. Modeling the costs of software quality. *Digital Technical Journal*, 5(4):9–16, 1993.

[15] H. Krasner. Using the Cost of Quality Approach for Software. *CrossTalk. The Journal of Defense Software Engineering*, 11(11), 1998.

[16] M. Krishnan. Cost, Quality and User Satisfaction of Software Products: An Empirical Analysis. In *Proc. 1993 Conference of the Centre for Advanced Studies on Collaborative research*, pages 400–411. IBM Press, 1993.

[17] M. Krishnan. *Cost and Quality Considerations in Software Product Management*. Phd thesis, Carnegie Mellon University, 1996.

[18] S. Kusumoto. *Quantitative Evaluation of Software Reviews and Testing Processes*. PhD dissertation, Osaka University, 1993.

[19] S. Kusumoto, K. 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.

[20] B. Littlewood, P. 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.

[21] C. Lott and H. Rombach. Repeatable Software Engineering Experiments for Comparing Defect-Detection Techniques. *Empirical Software Engineering*, 1(3), 1996.

[22] Y. Malaiya, A. von Mayrhauser, and P. Srimani. The Nature of Fault Exposure Ratio. In *Proc. 3rd IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pages 23–32, 1992.

[23] W. Mandeville. Software Costs of Quality. *IEEE Journal on Selected Areas in Communications*, 8(2), 1990.

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

[25] M. Paulk, C. Weber, B. Curtis, and M. Chrissis. *The Capability Maturity Model: Guidelines for Improving the Software Process*. The SEI Series in Software Engineering. Addison Wesley Professional, 1995.

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

[27] A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, and T. Stauner. One Evaluation of Model-Based Testing and its Automation. In *Proc. 27th International Conference on Software Engineering (ICSE'05)*. ACM Press, 2005.

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

[29] I. Sommerville. *Software Engineering*. Addison-Wesley, 7th edition, 2004.

[30] S. Wagner. Efficiency Analysis of Defect-Detection Techniques. Technical Report TUMI-0413, Institut für Informatik, Technische Universität München, 2004.

[31] S. Wagner and J. Jürjens. Model-Based Identification of Fault-Prone Components. In *Proc. Fifth European Dependable Computing Conference (EDCC-5)*, volume 3463 of *LNCS*, pages 435–452. Springer, 2005.

[32] 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.

[33] S. Wagner and T. Seifert. Software Quality Economics for Defect-Detection Techniques Using Failure Prediction. In *Proc. 3rd Workshop on Software Quality (3-WoSQ)*. ACM Press, 2005.