Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion

Bernd Freimut, Lionel C. Briand , and Ferdinand Vollei

Abstract: There is a general agreement among software engineering practitioners that software inspections are an important technique to achieve high software quality at a reasonable cost. However, there are many ways to perform such inspections and many factors that affect their cost-effectiveness. It is therefore important to be able to estimate this cost-effectiveness in order to monitor it, improve it, and convince developers and management that the technology and related investments are worthwhile. This work proposes a rigorous but practical way to do so. In particular, a meaningful model to measure cost-effectiveness is proposed and a method to determine the cost-effectiveness by combining project data and expert opinion is described. To demonstrate the feasibility of the proposed approach, the results of a large-scale industrial case study are presented and an initial validation is performed.

Index Terms: software inspection, cost-effectiveness model, MC-simulation, case study, expert opinion.

1 Introduction

It has been long recognized that the return on investment of software process improvement and software development technologies need to be assessed [31]. This entails the development of models that take into account both the costs of using a technology and its benefits across the whole development life-cycle in a realistic manner. Such models must not only represent accurately the economic impact of a technology but must also be usable at reasonable cost. Models based on parameters that cannot be estimated without resorting to prohibitive and complex data collection are not useful, from a practical standpoint.

The systematic, rigorous, and documented inspection of life cycle artifacts, before any testing can be performed, has long been perceived as a key software development technology to achieve high quality at a reasonable cost. Despite clear success stories [1], inspections do not bring substantial benefits under all circumstances. It is clear that, like for any technology, mostly success stories are selectively reported in the literature. Therefore, the cost-effectiveness of inspections needs to be demonstrated in each and every organization that tries to introduce them.

The problem is even more acute when considering that inspections usually encounter fierce resistance as they are introduced. For example, [15] report about the “not applicable here” syndrome and how, through experiments, they convinced developers to use inspections. They are usually supported by management but are sometimes perceived as burdensome by developers. This is due to the fact that inspections are often regarded as uncreative or unproductive work. For example, [26] reports that inspections are often perceived as “low tech, labor intensive, and rarely fun”. Moreover, in most organizations, productivity is still perceived and measured in terms of lines of code per effort unit. However, inspections require effort without producing any code. Consequently, inspections are sometimes not perceived as bringing tangible benefits. Studies [19] have shown that, despite having demonstrated their potential benefits for more than two decades, inspections are still not widely used across the software industry. Therefore, the case of systematic inspections needs to be made in every organization that intends to use them.

Furthermore, inspection benefits must be observed and communicated. The benefits of inspections consist on one hand of qualitative aspects, such as improving learning and communication, or serving as a means to enforce documentation standards. On the other hand, the benefits consist of quantitative aspects, such improved quality and effort savings. While the qualitative benefits are clearly an important asset of the inspection process, its ability to reduce development effort is often seen as key benefit [9].

One useful way to quantify the benefits of inspections is to assess the cost-effectiveness of inspections if one wants to be able to monitor the factors that affect it, its variations across inspections and projects, and then improve it. This implies a new requirement: cost-effectiveness models must allow for meaningful comparisons across inspections and projects.

The contribution of this paper is fourfold. First, a new model for inspection cost-effectiveness is presented. Second, a practical methodology for assessing inspection cost-effectiveness is presented that combines project data and expert estimates within a rigorous framework, using well-defined procedures. It is important to point out that our approach can be applied in actual project conditions as only data that can realistically be collected in that context are required. There is, therefore, no additional cost incurred besides common project data collection and interviews. Third, the result of a comprehensive case study is presented where a cost-benefit analysis of the inspection practices of a large Siemens AG division working on mobile communication was performed. Fourth, the proposed model and its application using expert estimates are assessed in terms of feasibility, consistency, and accuracy. Lessons learned are then drawn and discussed.

The remainder of the paper is structured as follows: Section 2 presents models for defining inspection cost-effectiveness as identified in a literature survey and discusses the requirements of these models. Section 3 presents a new inspection cost-effectiveness model that is obtained by generalizing and extending a model proposed by Kusumoto [20]. Since we suggest using expert estimates as surrogate for project data, we discuss in Section 4 the usage of expert opinion and present in Section 5 an approach to determine cost-effectiveness by combining project data and expert opinion. Section 6 presents the results of a case study, in which the proposed model and approach have been used. Section 7 presents the validation of the models and approach and Section 8 draws lessons learned. Section 9 concludes the paper.

2 Related Work

2.1 Modeling Cost-Effectiveness

In order to evaluate the benefit of software inspections and to motivate their dissemination various models have been developed that aimed at capturing the benefit of inspections. We surveyed the literature and identified a number of models that addressed various evaluation aspects such as inspections' efficiency, return on investment, and cost-effectiveness.

A simple model is the efficiency model proposed by Gilb and Graham [9] as

$$\text{Efficiency} = \frac{\text{Number of defects}}{\text{Cost consumed by inspections}} \quad (1)$$

This model defines the economic impact as the number of defects detected per time unit. Thus, it captures how effective is the inspection per unit of effort. The model only depends on data that is available after the inspection meeting and can therefore be easily determined once an inspection has been completed. On the other hand it does not compare the defect detection in inspections with defect detection in subsequent verification phases (i.e., savings are not addressed with this model).

A second model defines the economic impact of inspections in terms of the ratio between the cost and the benefit measured as effort saved. Rico [27] denotes this model as Benefit-Cost Ratio Model and Collofello and Woodfield [5] as Cost-Effectiveness Model.

$$\text{CE} = \frac{\text{Cost saved by inspections}}{\text{Cost consumed by inspections}} \quad (2)$$

An operational definition for this type of model is proposed by Collofello and Woodfield [5]. In order to estimate the savings from inspection their model assumes that defects removed from an artifact through inspections would have been detected and removed in a later defect detection phase. Therefore, the potential costs of detecting and correcting defects in these later phases are saved by inspections. The costs of detecting and correcting defects can be calculated as the average effort to detect and fix a defect times the expected number of defects detected. The expected number of defects detected by process can be calculated as the defect detection effectiveness of the process times the numbers of defects from process which still have not been detected. This procedure implies the tacit assumption that one defect in phase i will result exactly in one defect in phase i+1 (e.g., one defect in the specifications will result in one defect in the design).

A third model defines the economic impact of inspections in terms of return-on-investment (ROI) [11], [8], [27]:

$$ROI = \frac{\textit{Cost saved by inspections - Cost consumed}}{\textit{Cost consumed by inspections}} \quad (3)$$

In contrast to model (2) this ROI model only considers the net savings of inspections, which are computed by subtracting the cost consumed from the cost saved.

The fourth model has been proposed by Kusumoto et al. [20] and defines cost-effectiveness (CE) as

$$\text{CE} = \frac{\text{Cost saved by inspections - Cost consumed}}{\text{Potential defect cost without inspections}} \quad (4)$$

This model is similar to the ROI model (3), except that the denominator is not the inspection cost incurred by the project but the potential cost of defects that would have resulted if the detection activity to be assessed (i.e., inspections) had not taken place. Due to this normalization, cost-effectiveness can be compared across inspections as it is independent from the particular defect population present in the inspected artifact and the subsequent defect detection activities [20]. Model (4) was also extended by Sabaliauskaite [30] to account for false positives, meeting loss and meeting gain.

The model is intuitive as it calculates the net savings of inspections and compares them to the maximum defect-related cost that could have been saved. The CE value can therefore be interpreted as the percentage of defect–related cost that is saved due to inspections. However, this model also assumes that that one defect in phase i will result exactly in one defect in phase i+1.

2.2 Discussion

In order to assess the cost-effectiveness of inspection processes, it is necessary to have a model that operationally defines how to measure cost-effectiveness. This cost-effectiveness model must capture information relevant for economic decisions and for motivating inspections. Furthermore, it must be applicable when comparing cost-effectiveness across projects or project phases so as to help identify the kinds of inspections that need to improve and the contexts in which they are most effective. Consequently, the cost-effectiveness model that is required in our context must meet the following requirements:

- R1 – Savings: Cost-effectiveness must be stated in terms of effort savings, in order to make it relevant to quality assurance engineers and managers.

- R2 – Detection Activities: The cost-effectiveness model assumes a number of defect detection phases following inspections and can be tailored to a specific software development life cycle.
- R3 - Comparability: Cost-effectiveness can be compared across projects and project phases, regardless of the number of defects present in the inspected artifact and the number of subsequent defect detection activities taking place.

In addition to these criteria, we want to relax the assumption of the models proposed by Collofello, Woodfield, Grady, van Slack and Kusumoto, that one defect in phase i will result exactly in one defect in phase i+1. We found it necessary as our observations in industry shows that this assumption might not hold in all contexts [4]. Therefore we define:

- R4 – Defect Propagation: The cost-effectiveness model assumes that a defect detected by inspection will propagate in one *or more* defect in a later defect detection phase.

In order to select an appropriate model for our context, we compared the four models presented above to the proposed criteria. Table 1 reports on the score of each model with respect to each requirement. The scores are "+", "-", and "(-)". A "+" is assigned when the requirement is met, a "-" if it is not met, and a "(-)" if the requirement is not met but has been addressed by others. The score "n/a" is assigned if the requirement is not applicable.

	Gilb (1)	Collofello (2)	Grady (3)	Kusumoto (4)
R1: Savings	-	+	+	+
R2:Detection Activities	-	+	(-)	(-)
R3:Comparability	-	-	-	+
R4: Propagation	n/a	-	-	-

Table 1: A Comparison of CE Models

With respect to the first requirement, the Efficiency model by Gilb and Graham takes only into account the costs of performing inspections but not any potential savings in subsequent activities. Thus, it does not fulfill R1. The models by Collofello and Grady fulfill this requirement as they determine Return on Investment and Cost-Effectiveness, respectively. Thus they are stated in terms of effort savings. The cost-effectiveness model by Kusumoto calculates the net savings of inspections and compares these net-savings to the potential defect cost that would have incurred without inspections. The cost-effectiveness value can therefore be interpreted as the percentage of defect rework costs that are saved due to inspections. With this definition, cost-effectiveness is also stated in terms of effort savings.

With respect to the second requirement, the model by Gilb and Graham does not consider any subsequent activities. Thus, R2 is not fulfilled. The model by Collofello/Woodfield is defined in general for any number of defects detection activities and therefore fulfils R2. The models of Grady/vanSlack and Kusumoto consider only two detection activities, namely inspections and testing. However, using the principles of the Collofello/Woodfield model, they can also be generalized to any number of defect detection activities [2], hence their “(-)” score for R2.

With respect to the comparability across projects, the Gilb/Graham model does not fulfill R3, since variations in initial document quality (i.e., defect density) will make efficiency impossible to compare across inspections [16]. The models proposed by Collofello/Woodfield (2) and Grady/van Slack (3) might be problematic if we attempt to compare their results across projects. The following example illustrates why this is so. Suppose two projects with two defect detection activities: inspections and testing. Assume further, that in both projects, if inspections had not been performed, the costs of testing would be 1000 units. The first project consumes 10 cost units for their inspections and saves 100 units. Thus, the total cost for defect detection is 910 units. In the second project inspections cost 60 units and save 600. Thus, the total cost for defect detection is 460 units, which is far smaller than the cost in the first project. However, in both projects a ROI of 10 would have been computed, which would prevent us from recognizing the economic advantage of inspections in the second project. Consequently, these models do not fulfill R3. The model proposed by Kusumoto normalizes the estimated savings by the potential defect cost. Hence, it can be compared across different types of inspections (e.g., among inspections in different projects or different phases of the life-cycle). Thus, this model allows a meaningful comparison across different inspections and projects, thereby fulfilling R3.

With respect to the assumption of defect propagation, this requirement is not applicable for the Gilb/Graham model as this model does not consider any subsequent defect detection activities. The models proposed by Collofello/Woodfield, Grady/vanSlack, and Kusumoto assume that one defect in phase i will result in exactly one defect in the following phase i+1. Thus, neither of these models fulfils R4.

In summary, the model proposed by Kusumoto is closest to our requirements. However it assumes that one defect in phase i will result in exactly one defect in the following phase i+1. Moreover, it does not explicitly consider different phases in which inspections are performed and compared. Therefore, this model needs to be generalized to multiple inspection activities and its assumptions regarding defect propagation need to be relaxed.

3 An Extended Cost-Effectiveness Model

3.1 Model Definition

To define an inspection cost-effectiveness model fulfilling all our requirements, we started with the model proposed by Kusumoto. Generally, this model defines cost-effectiveness (CE) as the net savings from inspections over the potential defect cost without inspections as shown in formula (4).

The *Potential Defect Cost without Inspections* parameter is the defect rework cost that would have been incurred if no inspections had taken place. For example, suppose a life cycle with two defect detection activities: code inspections and testing and a source code listing with 100 defects. The potential defect cost is the rework cost of fixing all 100 code defects not in code inspections but during testing or after delivery. The *Cost consumed by inspections* parameter is the cost spent on performing code inspections, while *Cost saved by inspections* captures the cost saved in later phases due to defect detection in code inspections.

To generalize this model to multiple inspection activities, we assume an ordered set of defect detection activities and estimate initially the inspection savings using the approach of Collofello and Woodfield [2].

To relax the defect propagation assumption, we suggest to introduce a defect propagation factor, that captures the average number of defects in which defect that is not detected will result. Table 2 shows the parameters of the resulting extended cost-effectiveness model. In the remainder of the paper this model will be referred to as the CE model (for Cost-Effectiveness).

Para-meter	Description
A	Sequence of defect detection (inspection or testing) activities: [a_1, …, a_n]
$\bar{d}\varepsilon_i$	Average defect rework cost in activity *a_i* For inspections, these costs include the correction of defects. For tests, these costs include the isolation of faults from failures, correction of faults, and regression testing.
$\bar{i}\varepsilon_f$	Average inspection cost per defect in activity *a_f* (i.e., the cost per defect for preparing and performing inspections)
n_i	Number of defects found in activity *a_i*
N_i	Total number of defects present in activity *a_i*
$p_i=n_i/N_i$	Effectiveness of defect detection activity *a_i*. This parameter can also be seen as the probability to find a defect in detection activity *a_i*.
$\bar{g}_{i,i+1}$	A defect found in activity *i* would have resulted on average in $\bar{g}_{i,i+1}$ defects in the following defect detection activity (the original Kusumoto assumed this value to be equal to 1 for all *i*)
$r_{i,j}$	Assuming we assess the cost-effectiveness of inspection activity *a_i*, this is the remaining proportion of defects detected in *a_i* that would reach activity a_j had *a_i* not taken place

Table 2: Parameters of CE model

Using these parameters, the CE model can be formally expressed as follows:

$$CE(a_f, A) = \frac{\sum_{i=f+1}^{n} \bar{d}\varepsilon_i \times n_f \times r_{f,i} \times p_i - ((\bar{d}\varepsilon_f + \bar{i}\varepsilon_f) \times n_f)}{\sum_{j=f+1}^{n} \bar{d}\varepsilon_j \times N_f \times r_{f,j} \times p_j} \quad (5)$$

with

$$r_{f,j} = \begin{cases} 1 & j = f+1 \\ \prod_{k=f+1}^{j-1} (1-p_k) \times \bar{g}_{k,k+1} & else \end{cases}$$

In this model the sum $\bar{d}\varepsilon_i \times n_f \times r_{f,i} \times p_i$ denotes the *Cost saved by inspections*. This is obtained by summing up the cost saved in all defect detection activities following the inspection activity *a_f* to be assessed. The cost saved for a subsequent defect detection activity *a_i* is obtained by multiplying the average effort to find and fix a defect in *a_i* with the estimated number of defects in *a_i* that were prevented by inspections *a_f*. The term $(\bar{d}\varepsilon_f + \bar{i}\varepsilon_f) \times n_f$ denotes the *Cost consumed by* *inspections*, i.e., the effort spent on the inspections in a_f. These costs are subtracted from the cost saved to obtain the net savings. Finally, the sum $\bar{d}\varepsilon_i \times N_f \times r_{f,i} \times p_i$ denotes the *Potential defect cost without inspections*, which is computed in a similar way to the cost saved.

3.2 Model Assumptions

The proposed model has underlying assumptions that have to be assessed to determine whether they hold for the context under study. A minimal set of assumptions is, however, necessary to obtain a cost-effectiveness model that can be operationalized under realistic constraints. These assumptions are for the CE model:

1. The effectiveness of the last defect detection activity equals 1, i.e., all defects are eventually detected.
2. A defect found in activity a_i would result, on average, in $\bar{g}_{i,i+1}$ defects in the following defect detection activity $a_{i+1.}$
3. Defects found in inspections would, on average, result in the same cost in later defect detection activities as defects that slipped through inspections.
4. Defects introduced after the defect detection activity to be assessed must not be considered in the CE model for that activity.

The practical implication of the first assumption is that all defects will eventually be found by the last defect detection activity. In reality this activity is the operation of the software. But in order to determine the inspection cost-effectiveness after completion of the project and to provide timely feedback on the cost-effectiveness of a project's inspections, the last activity can also be set to be the last testing activity, e.g., acceptance testing. We then assume that most defects have been found by that time and that the effect of such assumption is therefore negligible. In the worst case scenario where this is not the case, the model provides a conservative estimate of the actual savings.

The second assumption implies that only defects causing a fault or failure in subsequent inspection or testing activities are considered. In the context of inspections, defects are typically classified as major and minor defects [9]. Major defects are those that would result in a defect in subsequent activities and other defects are denoted as minor defects. Therefore, only major defects should be used to determine the model's parameters.

The third assumption implies that, from a cost point of view, there is no difference in the defects that are detected by inspections and those that were *not* detected by inspections. Depending on the particular project this assumption might not hold. Therefore it is important to check the applicability of this assumption with experts.

To fulfill the fourth assumption, two practical implications have to be taken into account when instantiating the model for a particular sequence of defect detection activities. First, for each inspection activity, a specific model has to be instantiated. For example, to assess and compare analysis, design and code inspections, three models need to be developed. Second, the origin of defects (i.e., analysis, design, or code) has to be determined in order to compute the model parameters for each detection activity.

For example, when assessing the cost-effectiveness of design inspections, only analysis and design defects should be used to estimate the parameters. Thus, instead of determining the overall number of defects detected in a particular defect detection activity (e.g., design inspections, code inspections or testing), the number of analysis and design defects must be determined. Similarly, instead of determining the average defect cost in a defect detection activity, only the average defect cost for analysis and design defects has to be determined. In general, when instantiating the CE model for analysis, design, and code inspections, its parameters have to be determined for each defect origin separately.

To determine our model's parameters it is necessary to collect data on defect origins and rework effort throughout the development process. However, a survey performed among German software companies reported that less than 6% and 30% of the companies collect data on rework and defect origins, respectively [21]. Thus, although it is preferable to use consistent and reliable data sources to estimate the parameters of our model, it is very likely that, for some of these parameters, no data is collected in most industrial environments. In addition, information regarding the propagation factor $\overline{g}_{i,i+1}$ is also typically not collected in most of the organizations we have worked with.

To cope with this situation we propose a cost-effectiveness assessment procedure that uses expert estimates as a substitute for missing project data. In order to apply this approach, it is necessary to express the model in terms of parameters that can either be easily obtained through data collection during project performance or through the elicitation of expert opinion.

4 Using Expert Opinion

There are many reasons why expert opinion may be needed. One of the common problems, that we face here, is when information regarding a phenomenon cannot be collected by any other affordable means (measurements, observations, experimentation) or when the required data is simply not being collected in the timeframe CE is investigated. When looking at the problem from a scientific perspective, an important question is whether expert data is valid data.

It might be argued that expert data is soft data in the sense that they incorporate the assumptions and interpretation of the experts [22]. Specifically, expert data is subject to bias, uncertainty, and incompleteness. However, these problems can be prevented and controlled by means of carefully performed elicitation of expert estimates. Thus, expert judgment can be defined as data gathered formally, in a structured manner, and in accordance with research on human cognition and communication [22].

Therefore, we believe that expert judgment is valid data in the sense that it is comparable to other types of data. Expert judgment has also been used in similar ways, and with success, in other fields such as nuclear engineering (risk models) [13] and policy decision making [29], and in software engineering for cost estimation purposes [14] [3][17] and quality evaluation [28].

In order to prevent the problems mentioned before, specific expert knowledge elicitation techniques can be applied. The objective of these techniques is to prevent bias, uncertainty and incompleteness to the maximum extent possible, to detect bias when it occurs, and to model uncertainty. Elicitation techniques achieve this by carefully selecting experts, designing the mode of data collection and the interview procedure, and quantifying uncertainty in experts' responses.

One important aspect regarding elicitation concerns the *disaggregation* of the complex concept of cost-effectiveness. While it might seem straightforward to directly ask developers to estimate inspection cost-effectiveness, such an approach is not feasible. In order to provide accurate estimates, the experts need to remember relevant information regarding the parameter to be estimated [22]. Since the definition of cost-effectiveness (i.e. proportion of rework cost saved due to inspections) is rather abstract and its main parameters (*Cost Saved by Inspections* and *Potential Defect Cost without Inspections*) cannot be observed, the experts have no empirical experience on which they can base their estimates for such parameters. Therefore, it is necessary to disaggregate cost-effectiveness into parameters that the experts could have observed and experienced. This approach is consistent with studies in expert knowledge elicitation, which have shown that disaggregating complex questions can ease the burden of information processing for the expert and thus promote estimation accuracy [22].

In addition, we consider two issues that are particularly important in the design of the elicitation process: uncertainty and bias. These issues are discussed in the remainder of this section.

4.1 Uncertainty

Looking at the list of parameters in Table 2, there are three major kinds of parameters that are required by the model and that may need to be estimated by experts. One set of parameters concerns the effort for correcting defects of various origins in all defect detection activities ($\overline{d}\varepsilon_i$). The second set of parameters addresses the percentage breakdown of the various defect origins for all defect detection activities (p_i). The third set of parameters addresses the defect propagation factor $\overline{g}_{i,i+1}$.

It is impossible in an elicitation situation (e.g., an interview) to request a single value from the expert for these parameters. One reason is that subjective estimates are inherently uncertain. Providing an exact answer may not be possible since experts may not know the exact value of a parameter or this parameter value may actually vary with circumstances. For example, it is obvious that the cost of defect correction does not warrant a unique value but a probability distribution.

To explicitly capture this uncertainty a probability distribution can be selected as response mode. With such a distribution the experts are able to quantify their uncertainty. The probability distribution most often used in expert opinion elicitation is a triangular distribution as shown in Figure 1. Thus, the expert is asked to provide a range, given by minimum and maximum values, in which the estimate can be and the most likely value. This is shown in the left side of Figure 1.

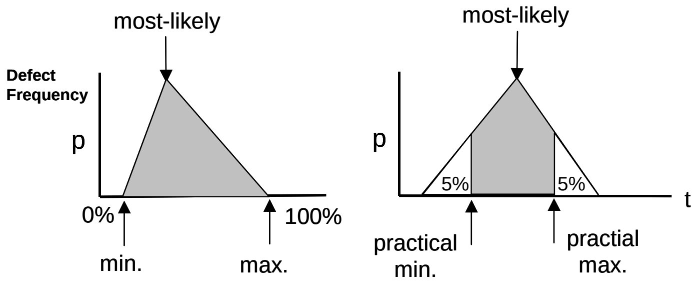

Figure 1: Triangular distributions

The use of a non-parametric distribution, such as the triangular distribution, is often recommended over parametric distributions in the domain of expert elicitation as their parameters have an intuitive appeal to the experts and are easy to respond to [32]. Moreover, the triangular distribution is often considered appropriate when little is known about the actual distribution of a variable. In the proposed approach this response mode is typically used to ask for the percentage of defects of a particular origin in a defect detection activity.

For the estimation of effort data, however, maximum or minimum values can be misleading. This is due to the fact that outliers (i.e., extreme low and high values that occur very rarely) can get excessive weight when using a triangular distribution [32]. Therefore, we ask here for *practical* maximum and minimum values. These are values that have some reasonable chance of occurring. To the experts the practical extreme values can also be explained by stating that the range between these values should contain about 90% of all possible effort values. This is illustrated on the right side of Figure 1.

To help experts visualize these response modes during the elicitation, slides similar to Figure 1 should be shown to them. Thus, they have a clear understanding of the information (e.g., maximum, minimum, most-likely values) they are to provide. To illustrate how uncertain information can then be actually captured during interviews, an excerpt from a questionnaire is shown in Figure 2.

Questions regarding the effort for fixing defects in design inspections

As a design author it is your task to fix defects detected in *design inspections*. This involves a decision, whether an issue is a real defect, finding a solution for the fix and changing the design accordingly. *How long does it typically take to find a solution for the fix and change the design accordingly?*

Suppose you determined several major defects that were introduced during design.

– In which range, according to your experience, can the effort for fixing one of these defects lie? ___ h to ___ h

– What would you estimate as a most likely effort for fixing one of these defects? __ h

Figure 2: Excerpt from Questionnaire

4.2 Preventing Bias

During elicitation, the experts will perform four cognitive tasks [Meyer and Booker, 1991]. The expert must comprehend the wording and the context of the question. Then he or she must remember relevant information to answer each question. By processing this information the expert identifies an answer, which is said to be “internal” as it is in the expert’s own representation mode. At this stage, people typically use mental shortcuts called heuristics to help integrate and process the information [Kahneman et al., 1982]. Finally, the internal answer has to be translated into the response mode requested by the interviewer.

In each of these steps, especially in applying the heuristics, systematic errors can occur, which would distort the estimate [Hofer, 1986]. To obtain reliable data it is therefore necessary to anticipate these biases and design and monitor the elicitation accordingly. This involves the following two activities in the design of the elicitation:

1. *Anticipate which biases are likely to occur in the planned elicitation.* For this step, the available research literature on expert opinion elicitation is used to determine which sorts of bias can occur and under which circumstances they can be expected [Meyer and Booker, 1991]. A list of potential biases identified as relevant for the approach is listed in Table 3.

2.

Bias	Definition	Cause
Wishful thinking	people's hopes influences their judgement	what should happen influences thinking
Inconsistency	people are inconsistent in their solving of problems	unintentional change of assumptions through fatigue, confusion
Over-confidence	people underestimate the amount of uncertainty in their answers	people think that being an expert implies the ability to give exact answers
Availability	people retrieve events with different ease from long-term memory	people do not receive scenarios triggering other, less accessible memory associations

Table 3: Bias definitions and sources in the case study

3. *Re-design the planned elicitation to make it less prone to the anticipated biases.* To do so, an initial draft version of interview questionnaires should be modified to account for the listed potential biases in Table 3. For example, it might be straightforward to simply ask the interviewees directly for maximum and minimum values of the parameters to be estimated. This, however, could introduce bias due to overconfidence. This type of bias occurs when experts intuitively minimize the uncertainty in their answers. To address this it is necessary to require the experts to think of possible scenarios under which a maximum or minimum value could occur. Thus, they process more information to provide an estimate for these values and this, based on existing research, typically results in better estimates. Also the order in which the estimates are elicited matters. Asking first for the most likely value might introduce bias due to overconfidence as experts tend to be anchored to their first estimate and fail to adjust appropriately for the range [Vose, 1996]. Consequently, the order of elicitation should be maximum, minimum, most-likely. As a result, the interview procedure looks as shown in Figure 3.

- Read introduction of question. Emphasize that we only refer to major defects.
- Ask whether there is a difference in correcting major and minor defects.
- Ask what are the typical activities that are to be performed to correct a major defect in design inspections. Propose other activities that were mentioned in previous interviews, if any.
- Ask whether the activities differ for different defect origins.
- Emphasize that we are interested in practical minimum and maximum values. Explain these values by means of the visualization of the response mode.
- Ask: From your experience, in which practical situations would the correction effort be very high? What would be then a practical maximum value for the correction effort?
- Ask: From your experience, in which practical situations would the correction effort be very low? What would be then a practical minimum value for the correction effort?
- Ask: Is the most likely value closer to the minimum or the maximum? What would you deem to be the most likely value?

Figure 3: Excerpt from Interview Procedure

A second important bias to address is inconsistency. This can occur if people get tired, forget information they were provided with, and therefore change assumptions or definitions in the course of the interview. To alleviate these issues, the questions should be phrased so that the key concepts and definitions are repeated in each question. Additionally, breaks should be planned in the interview procedure to prevent fatigue. Also, when the expert mentions scenarios, the interviewer should encourage the interviewee to consider additional, possible scenarios or, if possible, suggest additional scenarios. This addresses the bias of availability, which occurs when experts focus only on a few, recent scenarios.

5 A Cost-Effectiveness Assessment Procedure

The purpose of this section is to provide an overview of the cost-effectiveness assessment procedure. The procedure is illustrated in Figure 4. The rectangles denote activities while the circles denote products.

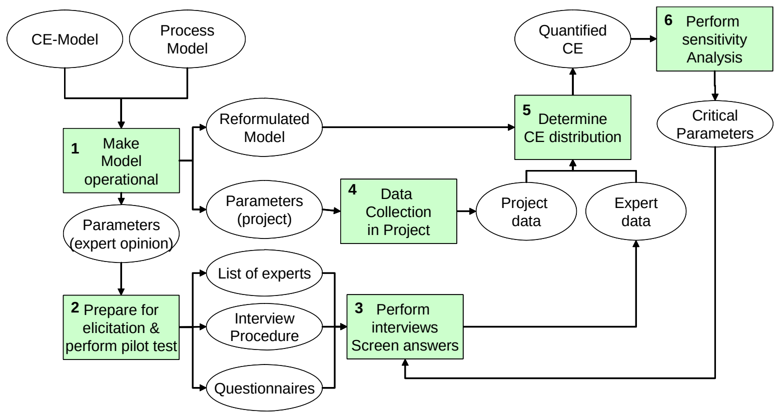
CE-Model
Process Model
Quantified CE
6 Perform sensitivity Analysis
Reformulated Model
5 Determine CE distribution
Critical Parameters
1 Make Model operational
Parameters (project)
4 Data Collection in Project
Project data
Expert data
Parameters (expert opinion)
List of experts
2 Prepare for elicitation & perform pilot test
Interview Procedure
3 Perform interviews Screen answers
Questionnaires

Figure 4: Overview of the assessment procedure

5.1 Step 1: Make Model Operational

The objective of the first step is to instantiate the CE model as defined in Section 3 for the projects under study. The CE model for a particular project must be instantiated for each inspection activity to be assessed (i.e., analysis, design, and code). Each of these models has to account for the detection activities following the inspection activity to be assessed and one therefore need to determine which activity is to be defined as last defect detection activity (cf. Section 3.2).

Since the tailored models include parameters whose values need to be determined, the next step is to decide whether this can be done based on available project data (e.g., metrics collected in inspection databases, testing databases, or defect tracking systems). If this is not the case, the parameter can perhaps be estimated by experts or can be reformulated as a function of variables that can be derived from project data or expert estimates. This process is called decomposition or disaggregation [23]. Disaggregation help ensure that expert-based parameters are in a form allowing the expert to concentrate on estimating something that is tangible and easy to envisage. Such parameters typically represent physical quantities, counts, proportions, and probabilities.

An example for disaggregation is illustrated by our case study example in Figure 5 (see Section 6 for details).

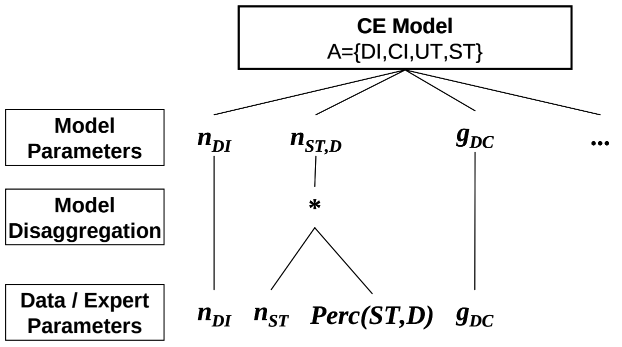

Figure 5: Disaggregation of CE Model (Example)

In this figure we assume that the CE model parameters encompass, among others, the number of defects detected in design inspections n_{DI}, the number of design defects detected in system test $n_{ST,D}$, and the propagation factor from design to code $g_{DC.}$ Typically, the number n_{DI} is available from inspection databases while g_{DC} needs to be estimated by experts. To obtain $n_{ST,D}$, the number of defects n_{ST} is usually available but not the defect origin information. In this case we multiply n_{ST} with the proportion of design defects detected in system test perc(ST,D), which can be estimated by testers.

As a precautionary measure, the assumptions of the CE model (cf. Section 3.2) also need to be checked in the context under study.

To summarize, this step of the assessment procedure yields 1) reformulated instantiations of the CE models (for each type of inspection), 2) a list of parameters to be obtained by expert knowledge elicitation, and 3) a list of parameters to be obtained from project data.

5.2 Step 2: Preparation for Expert Knowledge Elicitation

The objective of this step is to prepare the instruments of elicitation. These consist of 1) questionnaires that are used to capture the experts' estimates, 2) an interview procedure that guides the interviewer in performing the elicitation, and 3) the selection of experts.

Based on the output of Step 1, questionnaires have to be developed that capture the information to be obtained from the experts through well-formed questions [24]. For each parameter, a question similar to that of Figure 2 is integrated into the questionnaire to ensure that all necessary estimates are collected. Along with the questionnaire an interview procedure such as the one in Figure 3 is to be defined to guide the interviewer during the course of interviews. This aims at making the interviews more systematic and consistent and therefore obtains more reliable estimates.

Recall the aim of expert knowledge elicitation techniques is to reduce the impact of bias to the maximum extent possible and model the uncertainty in the experts' answers. In a literature survey on the use of expert opinion in risk assessment, Mosleh et al. [23] conclude that the methods by which expert opinion are elicited can have a significant impact on the accuracy of the resulting estimates. Thus, designing the questionnaire and interview procedure carefully and selecting appropriate response modes (i.e., the format in which the experts have to encode their answers) is of crucial importance.

Finally, in this second step, experts have to be identified, according to precise criteria, and motivated to do the job well. Several experts are necessary to estimate parameters as the multiplicity of answers will help cancel out random error [12].

The selection is guided by two criteria. The first one is the role of the experts in the development process. In their roles the experts must have access to the information they are supposed to estimate. For example, to estimate the proportion of defects of design origin in the total number of defects found in design inspections ($p_{DI,D}$), the experts must have participated in design inspections. Thus, designers and inspectors involved in design inspections qualify as experts. As another example, for the purpose of effort estimation, people performing the corresponding tasks qualify as experts. Thus, to estimate the effort required to fix a design defect, authors of the corresponding design documents would be selected as they perform the corrections. The second criterion for the selection of experts is their level of experience. Such experience needs to be sufficiently varied and extensive with respect to the targeted tasks.

In order to alleviate risks, the interview procedure and the questionnaire are usually tried in a pilot test. The purpose of these pilot tests is to gain feedback and optimize the elicitation accordingly. For example, we want to determine whether the experts are able to answer the questions and if there are sources of confusion and bias that might have been overlooked.

As a result, this step produces 1) validated questionnaires supporting the elicitation of the information identified in Step 1, 2) the corresponding interview procedure, and 3) a list of targeted experts.

5.3 Step 3: Performing Interviews and Screening Answers

Preparing the interviewers entails training them to acquire moderation and interviewing skills in general. Thus, the interviewers must know how to schedule, prepare, and conduct interviews. This encompasses skills such as how to structure the interview, what information to give to the interviewee, and how to create an open and constructive atmosphere during the interview. Such skills can be easily acquired using standard training courses and literature such as [22][24].

An interview is scheduled with each expert. Face-to-face interviews are preferable, as the experts are more motivated, and the interviewer has more control over the elicitation. However, depending on the amount of questions that have to be answered by the expert, one can resort to telephone interviews.

During the interview the interviewer guides the expert through the questionnaire using the precisely defined interview procedure. Appropriate visual aids should be used during interviews, especially those illustrating the response mode.

Moreover, during the interview, the elicitation has to be monitored for the occurrence of bias. Therefore, the following two activities are performed:

1. *Make the experts aware of the potential intrusion of bias*. Experimental studies in probability elicitation have shown that substantial improvement in the quality of assessments can be obtained through elicitation training [Hora and Iman, 1989]. The experts need to be informed about the biases they are likely to exhibit. In particular they should be informed about the definitions and causes of these biases [Meyer and Booker, 1991]. Therefore, the general concept of bias is to be introduced in the introduction of the elicitation interviews and the expected biases are presented. Additionally, the experts should be familiarized with the elicitation procedure, especially the response modes used in the questionnaire.
2. *Monitoring the elicitation for the occurrence of bias*. During elicitation, the interviewer

monitors the expert's body language and the verbalized thoughts of the expert. If these signs indicate some undesired situation, the interviewer should react accordingly. One example is that phrases like “we had a case...” could point out a potential outlier in the effort distribution. Whenever the expert mentions these phrases the interviewer should inquire about the representativeness of the case.

Once all interviews are completed, the answers of the experts are compared. If significant differences are observed this needs to be investigated further. If an expert’s answer differs significantly from that of other experts this may be explained by a different experience. In this case it must be assessed how representative and adequate this experience is.

This step produces, for each question (i.e., for each parameter in the model to be estimated by experts), a set of answers from the experts in the predefined response mode (i.e., the triangular distribution).

5.4 Step 4: Collect Project Data

The objective of the fourth step is to compile the project data in order to determine the values of those parameters for which it is required. This includes obtaining data from the appropriate sources such as inspection databases, testing databases, defect tracking systems, effort tracking systems and to estimate the required parameters. Thus, this step produces for each data-based parameter one single value.

5.5 Step 4: Compute Cost-Effectiveness

The objective of the fifth step is to determine the cost-effectiveness of inspections. Here the challenge is to combine the expert estimates collected in Step 3 with the project data collected in Step 4 to determine the cost-effectiveness according to the CE model reformulated in Step 1. The procedure for this is shown in Figure 6.

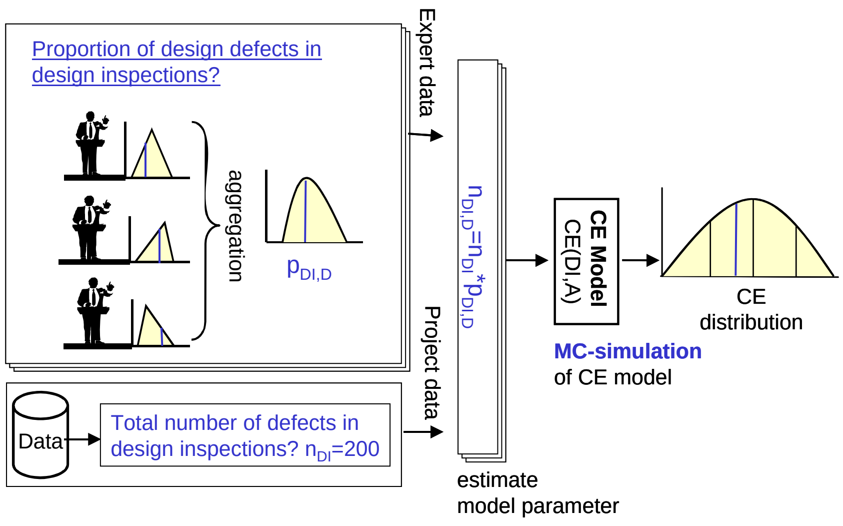
Proportion of design defects in design inspections?
aggregation
$p_{DI,D}$
Expert data
Project data
Data
Total number of defects in design inspections? n_{DI}=200
$n_{DI,D}=n_{DI}*p_{DI,D}$
estimate model parameter
CE Model
CE(DI,A)
MC-simulation of CE model
CE distribution

Figure 6: Steps in CE computation

For those parameters that are estimated by expert knowledge elicitation it has to be taken into account that several estimates are available for each parameter. Consequently the estimates of different experts, for each question, must be aggregated.

We must also consider that expert estimates are actually probability *distributions*. Consequently the resulting cost-effectiveness is also a probability *distribution* that needs to be computed. This distribution captures the inherent uncertainty in the cost-effectiveness of inspections. This uncertainty has two sources: the uncertainty in the experts' estimates and the inherent variation of cost-effectiveness across inspections. An important methodological point in presenting the results of any expert opinion study is to make explicit the underlying uncertainty of the results [13]. A probability distribution of the cost-effectiveness addresses this need.

Monte-Carlo (MC-) Simulation is a convenient way of performing the aggregation of experts' data [32] and the computation of the cost-effectiveness distribution. During one simulation run, a value for each input parameter is sampled from the experts' probability distributions. The set of sampled values forms a possible scenario, which is used as input to the model to compute the corresponding cost-effectiveness value. Repeating this procedure, say 1000 times, provides 1000 cost-effectiveness values, which form an empirical distribution. This procedure can be performed automatically using low-cost commercial tools such as @Risk [25].

MC-Simulation enables also to mathematically aggregate the estimated distributions. For example, the model requires the average effort to correct a defect. In the interviews we ask however for the effort distribution for correcting a single defect. In order to aggregate these estimates, we sampled one effort value from each expert's effort distribution and then take the average of all these sampled values. Proportion estimates are aggregated during the MC-Simulation by selecting one expert per simulation run and parameter (with all experts having equal probabilities of being selected) and sampling from the probability distribution of the selected expert.

5.6 Step 6: Sensitivity Analysis

The objective of this step is to identify those expert-based parameters that have a strong influence on the estimated cost-effectiveness. These parameters are highly influential in terms of the model accuracy and should therefore be elicited with extreme care. In addition they show which parameters can be influenced in order to improve the inspection process and consequently the inspection cost-effectiveness.

Technically, a sensitivity analysis is performed by analyzing the statistical correlation of the output variable of the model (i.e., cost-effectiveness) and its input variables (i.e., the expert-based parameters). The Spearman rank correlation coefficient is computed between the estimated cost-effectiveness and randomly sampled values of the estimated parameter distributions: the higher the correlation, the more significant the parameter in determining cost-effectiveness. For parameters showing a high correlation it would be advisable to interview additional experts in order to increase the representativeness of the estimates. The results can also be used to motivate the collection of project data for these parameters in order to determine cost-effectiveness with higher accuracy in the future. Last, the accuracy of the estimation process for these parameters should be carefully checked using redundant project data or specific experiments.

As a result, this step provides a list of critical expert-based parameters for the estimation of inspection cost-effectiveness. Based on this list a decision might be taken to perform additional interviews and incorporate their results to refine the model.

6 Industrial Case Study

6.1 *The Case Study Context*

The methodology presented in this chapter was developed specifically for a development organization and validated in a carefully prepared case study. This work took place in a business unit of Siemens AG, Germany, which develops products and services for mobile communication and intelligent networks using CHILL and Assembler.

In this particular business unit software quality is a major issue, as the downtime of a communication network has major consequences. Hence, inspections are performed after each of the development phases (i.e., analysis, design, and coding) to ensure sufficient quality for all software artifacts. Depending on the artifact to be inspected, three different kinds of inspection techniques are applied. Using the first technique, the artifact is distributed to several inspectors who read the document and send their comments to the author. The second, standard technique is similar to the one described in [6]: inspectors use checklists to identify defects during preparation and hold a meeting to collect and consolidate defects from all inspectors. The third technique, used for critical documents, enhances the second technique by having more interaction between authors and inspectors on the content of the inspected document and by discussing how to prevent the detected defects in the future, as proposed in [9]. On average, 10 participants perform an analysis inspection, and 4 participants perform a design or code inspection.

In this environment where software quality is a major objective, most of the data recommended in [6], [9] is collected during inspections. This encompasses information like the number of participants, the size of the work product, the inspection effort, the number of defects found (classed into major and minor defects) and the estimated rework effort.

6.2 *Step 1: Make Model Operational*

In the case study, since inspections take place at the end of analysis, design, and coding, three models needed to be developed. For example, if considering design inspections, the sequence of detection activities in the case study consists of design inspections (DI), code inspections (CI), unit test (UT) and system test (ST). Thus A=[DI,CI,UT,ST]. The parameters in Table 2 have been added a subscript indicating the activity they refer to (e.g., DI, for design inspection) or the defect origin they account for (i.e., AD for analysis and design defects combined).

$$CE(DI,A)=\frac{\sum_{i\in\{CI,UT,ST\}} \bar{d}\varepsilon_{i,AD}\times n_{DI,AD}\times r_{DI,j}\times\frac{n_{i,AD}}{N_{i,AD}}-((\bar{d}\varepsilon_{DI}+\bar{i}\varepsilon_{DI})\times n_{DI,AD})}{\sum_{i\in\{CI,UT,ST\}} \bar{d}\varepsilon_{j,AD}\times N_{DI,AD}\times r_{DI,j}\times\frac{n_{i,AD}}{N_{i,AD}}}$$

with

$$r_{f,j}=\begin{cases} 1 & j=f+1 \\ \prod_{k=f+1}^{j-1}(1-\frac{n_{k,AD}}{N_{k,AD}})\times\bar{g}_{k,k+1} & else \end{cases}$$

To check whether the model assumptions held, we presented the cost-effectiveness model to the quality assurance team and discussed its assumptions and thus validated them. The first assumption, that all defects are detected by system test, is an acceptable approximation as the products delivered undergo substantial quality assurance and are of high quality. The second assumption (defect propagation) is taken care of as we elicit this parameter. The Siemens QA team emphasized that the defect propagation factor might be different from one. For example, a defect concerning the design could result in several defects concerning the interfaces between two components. This observation supports the necessity to relax the original model assumption. The third assumption (defects found in inspections cause on average the same cost of defects not found in inspections) was deemed acceptable. Finally, assumption four (defects introduced later must not be considered) is addressed by tailoring the models.

In the case study, like in other typical cases, the parameters of this model for design inspections (and similarly the parameters for analysis and code inspections) could not all be derived from the existing project data nor could such data be collected in the near future. For example, the model requires $n_{i,AD}$, the number of analysis and design defects detected in a_i. In most activities, however, only the total number of defects was collected. Thus, the experts had to be asked for the percentage breakdown of the defect origins in each activity a_i. With this information it was then possible to determine $n_{i,AD}$.

Similarly, the average defect effort spent on analysis and design defects $\bar{d}\varepsilon_{i,AD}$ is required. For inspections, only an overall estimate for the corrections of all defects is available. This estimate contains both the effort for major and minor defects and, moreover, does not distinguish between the different defect origins. Thus, the experts had to be interviewed about the “typical” defect cost for a (major) defect of a particular origin detected in each inspection activity a_i. The correction effort for each defect (along with the defect origin) was only available during unit test. However, this effort data did not consider the effort for isolating faults after a failure was detected and did not consider the time to test a correction. For system test, none of the required effort data was available. Thus, the experts had to be interviewed about this as well.

Moreover, the model requires effort information in terms of average effort (e.g., average defect rework cost). But it might be difficult for experts (developers, testers) to estimate average effort values though it should be easier to estimate the effort for correcting a single defect. The experts can assign a meaning to this effort, since they experience this effort every time they correct a defect. Therefore, they were asked about effort distributions for correcting a single defect. The parameters for the design CE model above are described in Table 4. Table 5, Table 6, and Table 7 indicate how parameters can be estimated or computed.

Parameter	Description	Data Source
$\bar{d}\varepsilon_{i,AD}$	*Average* defect cost for defects originating in analysis or design and detected in activity a_i	Expert opinion
$\bar{i}\varepsilon_i$	*Average* inspection costs in activity a_i	Project data
$n_{i,AD}$	Number of defects originating in analysis or Design and found in activity a_i	n_i is available as project data. Expert opinion is required to compute $n_{i,AD}$
$\bar{g}_{i,i+1}$	Propagation factor: a defect found in activity a_i would *on average* result in $\bar{g}_{i,i+1}$ defects in activity a_{i+1}	Expert opinion

Table 4: Parameters of the CE model for design inspections

To determine those parameters in Table 4 that are (partly) based on expert estimates, the information shown in Table 5 is elicited from the experts.

Parameter	Description
$d\varepsilon_{i,<origin>, e}$	Defect cost for one defect of <origin> in activity a_i as estimated by expert *e*
$p_{i, <origin>}$	The percentage of defects of <origin> in activity a_i
$g_{i,i+1, e}$	Propagation factor for one defect from a_i to a_{i+1} as estimated by expert e

Table 5: Information to be elicited from expert opinion

To provide an overview of the entire set of involved parameters, Table 6 shows all parameters, either elicited from experts or obtained from project data. The latter are shown in italics.

Parameters		# Defects	Breakdown Origin			Prop. Factor	Rework Effort			Insp. Effort
			A	D	C		A	D	C	
Detection Activities	Analysis	n_{AI}	$n_{AI,A}$			g_{AD}	$d\varepsilon_{AI,A}$			$i\varepsilon_{AI}$
	Design	n_{DI}	$n_{DI,A}$	$n_{DI,D}$		g_{DC}	$d\varepsilon_{DI,A,}$	$d\varepsilon_{DI,D}$		$i\varepsilon_{DI}$
	Code	n_{CI}	$n_{CI,A}$	$n_{CI,D}$	$n_{CI,C}$		$d\varepsilon_{CI,A,}$	$d\varepsilon_{CI,D}$	$d\varepsilon_{CI,C,}$	$i\varepsilon_{CI}$
	Unit Test	n_{UT}	$n_{UT,A}$	$n_{UT,D}$	$n_{UT,C}$		$d\varepsilon_{UT,A}$	$d\varepsilon_{UT,D}$	$d\varepsilon_{UT,C}$	
	Sys.Test	n_{ST}	$n_{ST,A}$	$n_{ST,D}$	$n_{ST,C}$		$d\varepsilon_{ST,A}$	$d\varepsilon_{ST,D}$	$d\varepsilon_{ST,C}$	

Table 6: Overview over all parameters

With this information, some parameters can be computed as shown in Table 7:

Para-meter	Description	Disaggregation
$n_{i,<origin>}$	Number of defects originating in <origin> and found in activity a_i	$n_{i,<origin>}=n_i * p_{i,<origin>}$
$\bar{d}\varepsilon_{i,A}$	Average defect cost for defects that originated in analysis and were detected in activity a_i	avg.($d\varepsilon_{i,A,1}$,..., $d\varepsilon_{i,A,E}$)
$\bar{d}\varepsilon_{i,D}$	Average defect cost for defects that originated in design and were detected in activity a_i	avg.($d\varepsilon_{i,D,1}$,..., $d\varepsilon_{i,D,E}$)
$\bar{d}\varepsilon_{i,AD}$	Average defect cost for defects originating in analysis or design and were detected in activity a_i	($n_{i,A}$* $d\varepsilon_{i,,A}$ +$n_{i,D}$* $d\varepsilon_{i,D}$) / ($n_{i,A}$+$n_{i,D}$)
$\bar{g}_{i,i+1}$	Propagation factor: a defect found in activity a_i would on average result in $\bar{g}_{i,i+1}$ defects in activity a_{i+1}	avg.($g_{i,i+1,1}$,..., $g_{i,i+1,E}$)

Table 7: Computation of parameters from expert data

These tables essentially represent the list of parameters to be obtained either from project data or by expert knowledge elicitation. The instantiated model for design inspections is built by substituting parameters with the disaggregation formula in Table 7.

The parameters in Table 5 can be estimated by one of three different roles in the development process: (1) Analysts, who can estimate the correction cost for defects detected in analysis inspections, (2) Testers, who can estimate the effort for isolating faults from failures, and (3) Developers, who can estimate effort and defects breakdowns for the remaining activities.

6.3 Step 2: Preparation for Expert Knowledge Elicitation

In this step we developed the interview procedure and questionnaire as indicated in Figure 2 and Figure 3. As we identified three different roles to be interviewed, we developed a questionnaire and procedure for each role. Each questionnaire/procedure was tested in a pilot test. To do so, we performed the elicitation as planned with a few interviewees and debriefed the experts with specific question regarding the clarity of questions, the usage of definitions, and their ability to answer the questions. Based on the feedback received, the case study material was then refined.

To cancel out random error, four to six experts were selected to estimate one parameter, depending on the model and parameter. In total, 23 experts were selected. In addition to selecting experts with appropriate roles, several years of experience in inspections or testing was set as a requirement to qualify as expert.

6.4 Step 3: Performing Interviews and Screening Answers

To collect the expert estimates, we performed seven face-to-face interviews with Developers as these had to estimate the majority of parameters and an actual meeting was desirable for this. The interviews were planned to last about two hours each including a break to prevent fatigue. With Analysts and Testers we performed additional 16 telephone interviews, which lasted 30 minutes each. We deemed that for these roles telephone interviews were acceptable, as only few parameters were elicited.

In order to screen the answers we adopted the following rationale: If an expert's answer differed significantly from other experts' answers and if that expert did not show relevant experience for the questions in the recent past, or showed difficulties with the questions during the interview, then the answer was left out for the remainder of the analysis. If the expert did not differ from other experts in terms of experience, he was simply debriefed over the phone to determine possible reasons for his different answers. Depending on the explanations provided to the authors, e.g., how representative was the expert's experience, it was then decided whether to include the answer in the analysis.

One analyst mentioned in the interview that he had not participated in analysis inspections for a long time. Since his estimates about the correction effort of defects detected in analysis inspections differed from the other analysts, we decided to exclude this estimate. Similarly, one developer provided estimates for the defect correction effort in code inspections that differed from other developers' estimates. When debriefed over the phone it turned out that he was responsible for a rather complex part of the system. Since we deemed his experience was not representative for the entire system we decided to exclude his estimates as well. Thus, in total, one estimate by an analyst and three estimates by a developer were not retained for analysis.

6.5 Step 4: Collect Project Data

The project data was provided by the quality assurance team as a set of MS-Excel datasheets containing the relevant aspects of the inspection database, and defect information from unit test. The required data-based parameters could then be easily computed.

6.6 Step 5: Compute Cost-Effectiveness

Following the approach depicted in Figure 6, we computed the cost-effectiveness for analysis, design, and code inspections. Figure 7 shows the results of the MC-simulation performed using the extended and generalized cost-effectiveness model. The left and right graphs show, respectively, the probability distributions for cost-effectiveness and the cost saved by inspections. Since these graphs represent probability distributions, the x-axes denote the range in which the model value lies while the y-axes assign each cost-effectiveness value a corresponding probability value so that the area under the distribution curves integrates to 1.

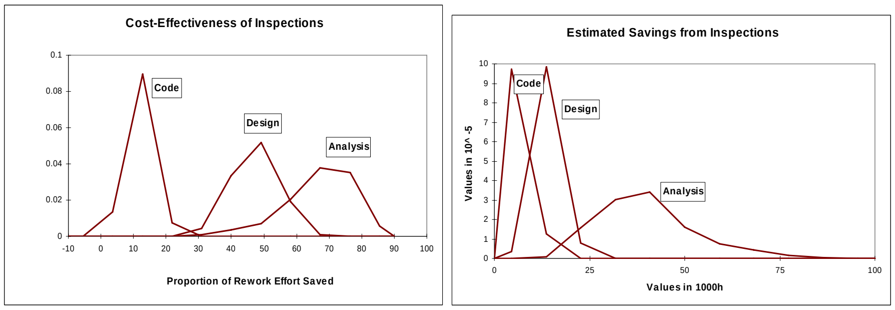

Figure 7: Case Study Results

The relevant statistics from these distributions are shown in Table 8 and Table 9. According to [31] the central tendency and variation range for variables such as savings or cost-effectiveness provide sufficient information for management decisions. Hence we report the minimum, maximum, mode, median, and mean for all three distributions.

	Min	Mode	Median	Mean	Max
Analysis	20	70	69	67	88
Design	30	50	47	47	67
Code	-1	13	12	12	24

Table 8: CE Statistics (in percent of saved rework)

	Min	Mode	Median	Mean	Max
Analysis	15946	39315	38435	39672	96803
Design	6416	15172	13559	13763	23432
Code	5385	8084	7859	7905	11170

Table 9: Savings Statistics (in 1000 person hours)

It can be seen that the distributions of cost-effectiveness as well as the savings for analysis, design, and code inspections are clearly ordered. Analysis inspections are considerably more cost-effective than their design counterpart, which are in turn markedly better than code inspections. Despite the uncertainty modeled during expert knowledge elicitation, the distribution patterns are quite clear with this respect. These results have important practical consequences. First, it can be seen here that earlier inspections, namely analysis or design inspections, have much higher benefits than code inspections.

Figure 7 suggests that the net savings of the three types of inspections follow the same pattern as cost-effectiveness. Such results help focus the QA team improvement efforts where the gain is likely to be more substantial. In this context code inspections exhibit lower savings and cost-effectiveness than analysis and design inspections. A more detailed analysis of the model parameters (cf. Table 2) indicated that this was due to a low effectiveness of the code inspection process (i.e., many defects slipped through). Consequently, a process improvement initiative should focus in priority on code inspections, as these provide most improvement potential to make them more cost effective and be economically viable. Furthermore, the QA team should investigate whether early inspections can be further disseminated throughout the development organization, as these yield high savings. In the case study, the QA team decided to focus future process improvement activities on code inspections, as they were satisfied with the cost-effectiveness of earlier inspections and did not see direct improvement potential. From a practical standpoint, it is interesting to note that getting tangible, quantitative evidence was key in triggering concrete actions.

In the future, the cost-effectiveness distributions can be used to evaluate future projects in order, for example, to assess whether a change in inspection procedures affects their cost-effectiveness. In other words, the distributions generated above could be used as baselines for future comparisons. Thus, changes to the inspection process can be assessed in terms of the improved cost-effectiveness.

6.7 Step 6: Sensitivity Analysis

The sensitivity analysis was performed using the built-in functionality of @Risk. The simulation provided 1000 samples for each expert-based input parameter. @Risk then computed the correlation between these sampled values and the related cost-effectiveness values. Table 10 shows the results for the analysis, design, and code models. For each model we report the most influential parameters (absolute correlation above 0.2), their descriptions, and the Spearman Rank Correlation between their values and cost-effectiveness values.

	Rank	Param.	Meaning	Correlation
Analysis	1	$\bar{d\varepsilon}_{AI}$	effort to correct an (analysis) defect in analysis inspection	-0,54
	2	$n_{AI,A}$	number of analysis defects in design inspections	-0,53
	3	$n'_{ST,A}$	number of analysis failures uncovered in system test	0,39
	4	$g_{A,D}$	Propagation Analysis to Design	0,31
	5	$g_{D,C}$	Propagation Design to Code	0,25
Design	1	$g_{D,C}$	Propagation Factor	0,61
	2	$n'_{UT,D}$	number of UT failures that originate from design	-0,42
	3	$n'_{UT,D}$	number of ST failures that originate from design	-0,37
	4	n_{CID}	number of design defects in code insp.	-0,28
Cod	1	$\bar{d\varepsilon}_{CI,D}$	Correction effort design defects in code inspection	-0.42
	2	$d\varepsilon_{isolate,UT}$	isolation effort for unit testing	0,21

Table 10: Sensitivity Analysis Excerpt

For both the analysis and the code model, the effort to correct a major defect in inspections ($\bar{d\varepsilon}_{AI}$, $\bar{d\varepsilon}_{CI,D}$) is the most significant parameter. These variables are one component of the *Cost consumed by inspections.* A high rework cost per defect will therefore increase the inspection cost, which in turn will decrease the cost-effectiveness.

The significance of $n_{AI,A}$ in the analysis model can be explained by the fact that a small number of analysis defects detected in design inspections will leave more analysis defects to be detected in later phases and will therefore increase the cost-effectiveness of subsequent inspections.

The two propagation factors from analysis to design ($g_{A,D}$) and from design to code ($g_{D,C}$) are among the influential variables in the analysis model. This could be expected as they are directly related to the number of defects in subsequent phases and therefore also the estimated savings and the estimated defect cost without inspections.

For the design model, the propagation factor $g_{D,C}$ is by far the most correlated with cost-effectiveness. But in the context of our case study, due to time constraints, it was not possible to perform additional interviews. However, as we will see in Section 7.4, the analysis of expert responses shows a high level of consistency for that parameter. The number of defects detected in unit testing ($n'_{UT,D}$) also shows also a strong correlation. Further analysis showed, as discussed in Section 7.3, that these estimates were sufficiently accurate.

7 Method Validation

In order to obtain supporting evidence that the instantiated models and their simulation results are sufficiently accurate, we have to resort to indirect means. This is due to the fact that the savings from inspection defect detection are not directly measurable; otherwise, we would not need expert estimates.

Such supporting evidence can take several forms. First, we can verify that the results are consistent with the intuition of testers, developers, and QA engineers. Second we can check whether our model's results are in line with previous research. Third, we can compare the results of our model with those of the original Kusumoto model with respect to the first two criteria, thus demonstrating if there is an improvement. When available, project data can be used to compute some of the parameters, and they can be compared to expert estimates. Finally, consistency among expert estimates can be analyzed and its impact on the simulation results can be assessed. Though none of these elements leads to a direct assessment of the models' accuracy, they provide information that can increase our level of confidence in the results.

This section not only provides a validation in our case study context. It also provides, through examples, a methodology for others to validate their own models.

7.1 How Plausible are the Results?

With respect to previous research results, the patterns shown in Figure 7 confirm what software engineering professionals usually acknowledge, i.e., early-artifacts inspections are probably more beneficial than code inspections [2] [10].

Furthermore, feedback sessions with QA engineers of the development organization confirmed that our results were consistent with their intuition. Based on previous analyses, they suspected the benefits of code inspections to be limited for some parts of the system. Our results confirmed this suspicion as the code inspection cost-effectiveness distribution tails near the 0 threshold (cf. Figure 7). Consequently, the QA team decided to improve the cost-effectiveness of code inspections in order to prevent common types of defects from slipping to the testing phase. They performed a causal analysis of those code defects slipping though code inspections and modified the code inspection checklists to address those types of defects.

7.2 Feasibility Analysis

In order to investigate whether experts are able to provide the required estimates, we used the pilot test in Step 2 of the procedure. Here several experts were specifically asked whether they felt it was possible for them to estimate the required parameters. The pilot testers unanimously agreed that, with their experience, all parameters in our CE models could be estimated, though for some parameters (e.g., defect rework effort for defects introduced early but detected late), this might be a bit more difficult. Also though our case study involved 41 different parameters and 23 experts, only in few instances an expert was not able to give an estimate.

7.3 Accuracy and Impact of Expert Estimates

To investigate how well the experts were able to estimate the parameters, we determined, when possible, the experts' estimation accuracy and its resulting impact on the model output. For this purpose we systematically compared, for selected model parameters, the expert estimates and the actual values determined from project data and investigated the impact on the model simulation output. Though in the environment under study this was only possible for a partial set of parameters, we hope these results are representative of the experts' general ability to estimate.

The first set of parameters we consider are the ones that capture the breakdown per defect origin for all defects detected in unit test (i.e., $p_{UT,A}$ $p_{UT,D}$ $p_{UT,C,}$ cf. Table 6). The second set of parameters concerns the effort for correcting unit test defects (i.e., $d\varepsilon_{UT,A}$ $d\varepsilon_{UT,D}$ $d\varepsilon_{UT,C}$). Thus, despite the limitations, we are at least able to investigate two of the main types of parameter estimates: defect proportions and task effort values.

7.3.1 Defect Origin in Unit Testing

How accurately do experts estimate the breakdown of defect origin for defects detected in unit test? To answer this question, we obtain the overall expert estimates for the corresponding model parameters by performing a MC-Simulation in order to aggregate all responses as described in Step 4 of the procedure. The resulting distribution means for the proportion of analysis, design, and code defects, respectively, are shown in Table 11. This table also shows the actual proportions determined from project data as well as the resulting difference and the relative error of the expert estimates.

Defect Origin	Estim. Proportion [%]	Act. Proportion [%]	Diff.	Relative Error
Analysis	6.09	8.67	-2.58	-33%
Design	24.50	17.01	+7.49	+25%
Code	71.92	74.32	-2.40	-4%

Table 11: Accuracy of expert estimates for defect origins in unit testing

It can be seen that the absolute differences between the estimated average proportion and the actual proportion range between 2.4% and 7.5%. The absolute relative error ranges between 4% and 33%. The question is whether such differences would have a significant impact on simulation results.

The cost-effectiveness probability distributions for analysis, design, and code inspections are shown in Figure 8. This figure shows the cost-effectiveness distributions derived from the available project data (solid line) and the cost-effectiveness distributions derived using the expert estimates for the defect breakdown per origin (dashed line).

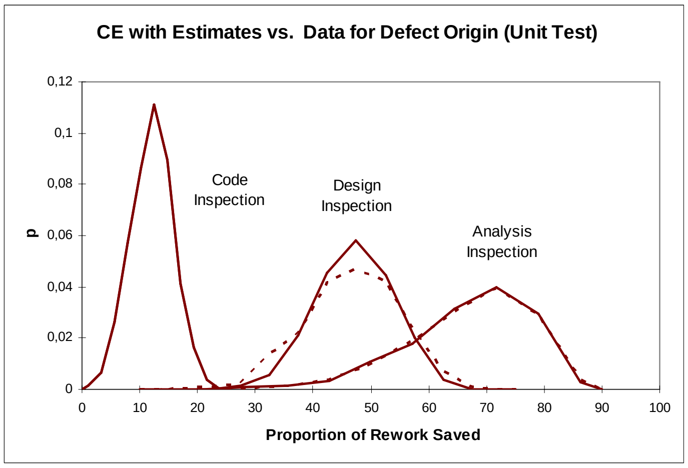

Figure 8 Comparison of model results with expert data and project data for defect origin proportions

It can be seen that for each model the differences between expert-based and data-based simulation results are very small, yielding almost identical distributions. As a consequence, similar conclusions regarding the inspections' cost-effectiveness would have been drawn from both distributions.

Therefore, we can conclude that, at least for the parameter under study, the use of expert estimates did not distort the results.

7.3.2 Correction Effort for Defects Detected in Unit Testing

How accurately do experts estimate the average effort required to fix a defect detected in unit test? The resulting distribution means of our comparisons with project data are shown in Table 12. This table shows the actual average effort as well as the resulting absolute and relative error of the expert estimates.

Defect Origin	Act. Avg Correction Effort [h]	Estim. Avg Correction Effort [h]	Diff.	Relative Error
Analysis	6h	9.97h	3.97h	66%
Design	3.9h	5.46h	1.56h	40%
Code	3.2h	3.19h	0.01h	0%

Table 12: Accuracy of Effort Estimates for correction effort in unit testing

Correction cost is highest for analysis defects and lowest for code defects, as would be expected. It can be seen that for code defects, the estimates are highly accurate, whereas for design and analysis defects, a relative error of 40% and 66%, respectively, indicates a larger overestimation. One possible explanation for this result is that more code defects are found in unit testing and, therefore, with the task of correcting a code defect being more frequent, this results in a better ability to estimate.

However, although the estimation error seems significant, a more important question is whether these errors would practically impact the simulation results. To investigate this let us look at Figure 9. Just like for the origin breakdown parameters, the differences between expert-based (dashed line) and data-based simulation results (solid line) are negligible.

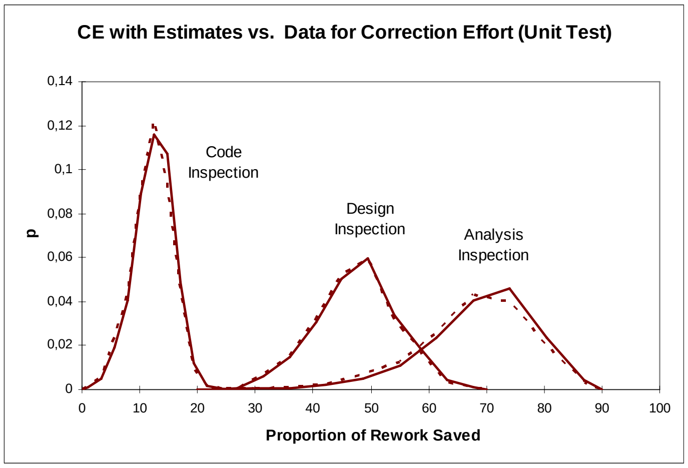

Figure 9: Comparison of model results with expert data and project data for correction effort

7.4 Consistency of Expert Estimates for Propagation Factor

Recall from Sensitivity Analysis that the propagation factors $\overline{g}_{i,i+1}$ have a strong influence on the simulation results of design and analysis CE models. Because no actual project data is available for these parameters as an indicator of their accuracy, we investigate how consistent the experts' estimates are. In Figure 10 the expert estimates for both propagation factors are shown.

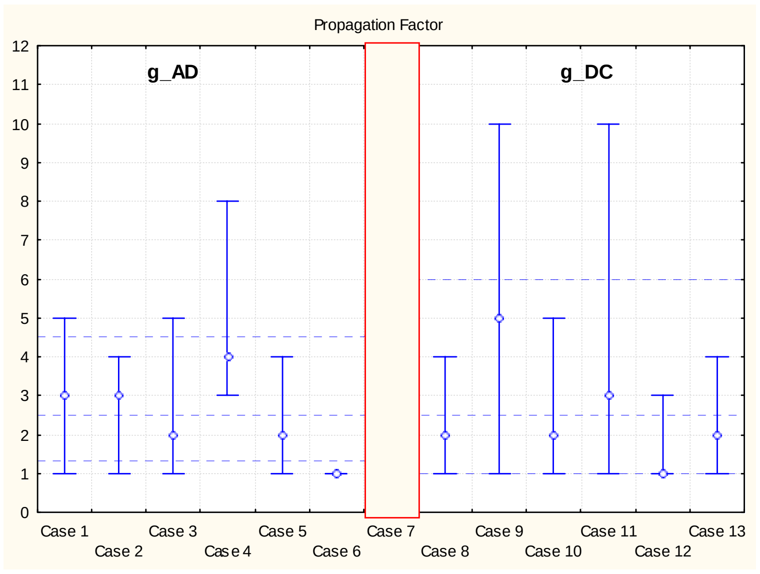

Figure 10: Consistency of expert estimates for propagation factor

This figure shows on the x-axis the individual experts and on the y-axis, for each expert, the estimated maximum, minimum, and most-likely values. Furthermore, the horizontal lines denote the means of the maximum, minimum, and most-likely values, respectively.

In Figure 10, we can see that the minimum and most-likely values are rather consistent across experts. However, two expert estimates show a significantly higher (practical) maximum value when compared to other experts. This might be due to the fact that all experts were responsible for different parts of the system.

To assess the impact of the propagation factor estimates of individual experts on cost-effectiveness, we followed a leave-one-out validation strategy. Thus, we performed six simulation runs, excluding one different expert in each, and re-computed cost-effectiveness. Because the number of experts involved is small and since multiple experts are necessary to represent the entire system and to cancel out random error, the leave-one-out strategy was considered the best option.

Table 13 presents for the analysis and design model the minimum, mean, and maximum values of the CE distributions when all experts are considered, or for each of the six additional simulations where one expert was excluded.

		Min	Mean	Max
Analysis	All experts	21	67	89
	Exclude Expert 1	22	67	90
	Exclude Expert 2	13	65	91
	Exclude Expert 3	25	68	89
	Exclude Expert 4	10	60	85
	Exclude Expert 5	21	69	91
	Exclude Expert 6	29	71	91
Design	All experts	23	47	65
	Exclude Expert 1	20	49	67
	Exclude Expert 2	21	43	64
	Exclude Expert 3	22	48	66
	Exclude Expert 4	23	44	62
	Exclude Expert 5	24	50	67
	Exclude Expert 6	22	49	67

Table 13: Impact of experts on CE

This table shows that overall, the distributions where one expert is left out are rather consistent with the distribution considering all experts. In most simulation runs the distribution means differ only by less than 4 percent. For min/max, values larger differences can sometimes be observed. These differences, however, can be attributed to the simulation procedure. When performing multiple simulations of the same model, we generally observed that min/max values could exhibit a larger variation than means.

Therefore, the results above confirm that by using a small group of experts, the estimation result of our simulation procedure is not overly sensitive to individual estimates.

7.5 Comparison between Original Kusumoto Model and its Extension

Another way to demonstrate that our extended model provides an improvement is to compare its simulation results to those of the original Kusumoto model. Are there differences? If yes, which results are more plausible?

We generalized the original model by assuming that a defect found in activity a_i results, on average, in $\bar{g}_{i,i+1}$ defects in the following defect detection activity a_{i+1}. Such a propagation factor is expected to significantly change the results of the model, as one defect can propagate to many defects in subsequent phases. Consequently, the effort saved due to inspections should be significantly higher with a propagation factor $\bar{g}_{i,i+1}$ >1 compared to a model where $\bar{g}_{i,i+1}$ =1.

Figure 11 shows that, for our case study, the cost-effectiveness probability distributions for both the original model and the extended model are significantly different. Moreover, the extended model provides, as expected, higher cost-effectiveness values than the original model.

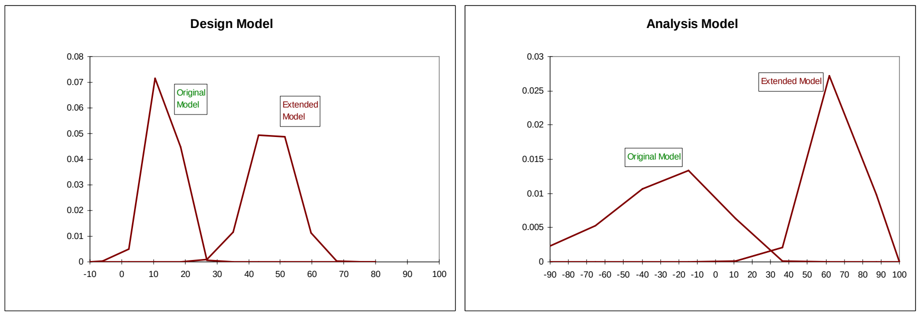

Figure 11: Differences between original and extended model

Unfortunately, it is not possible to draw straightforward conclusions regarding the accuracy of either model. This is due to the fact that savings from inspections are not observable and thus not directly measurable. Consequently, it is not possible to determine the actual cost-effectiveness in order to compare these with our two competing models. Therefore, we have to demonstrate the plausibility of our generalized model through indirect means.

The rationale for our comparison is based on the representation condition in measurement theory [7]: the CE model should map inspections so that empirical relationships are preserved by the numerical relationships. In other words, our simulation results should be consistent with intuition and the literature results in terms of what types of inspections are expected to be more cost-effective. In Section 7.1, the generalized model provided us with a clear ordering between distributions as shown in the right panel of Figure 7. Thus, analysis inspections are more cost-effective than design inspections, and these are, in turn, more cost-effective than code inspections. However, using the original Kusumoto model, analysis inspections are estimated to be less cost-effective than design or code inspections, while code inspections and design inspections are equally cost-effective.

To compare the plausibility of results across models, let us first look at the literature. There is ample evidence that inspections performed on early development artifacts are more beneficial than inspections performed later in the life cycle [2][10]. Therefore, we can conclude that the extended model provided results that are more plausible according to existing empirical evidence.

Another argument in support of the extended model's results is the ability to draw correct conclusions for the environment under study. The extended model quantitatively confirmed what was already the intuitive insight of the QA team: code inspections were considered problematic while design and analysis inspections were considered satisfactory. Based on the original Kusumoto model, different conclusions would have been drawn and this would have led to significantly different decisions.

To conclude, there is supporting evidence that the extended model provides results that are more plausible, therefore better reflecting the representation condition in our case study setting and the existing evidence in the literature.

8 Lessons Learned

During the performance of the interviews and the analysis of interview data, lessons learned were drawn that can roughly be divided into the following categories: motivation of experts, selection of experts, and the application of the elicitation techniques.

8.1 Motivation of Experts

Overall, in our case study, experts were highly motivated during the interviews. They showed great interest in the study and the kind of questions they were investigated. Very well received was the fact that the experts (i.e., the developers and testers) were involved in order to draw conclusions about the inspection practices in the business unit. In some cases, the experts expressed their satisfaction and interest to the QA team. The experts were also very interested in receiving a copy of the final case study report to see the final results of their efforts.

8.2 Selection of Experts

The QA management selected the experts following the rationale that they should be proactive and motivated developers and testers. This might also explain the positive atmosphere reported above. The experts had various levels of experience in the business unit ranging from 3 to 18 years. Some of the experts with 3 to 4 years of experience had difficulties answering some of the questions regarding relatively rare events, such as the correction effort for analysis defects that are detected late in the system testing phase. Thus, experts should have a sufficient level of experience to be able to answer the questions. In a particular context, this should be determined during pilot testing. Additionally, they should currently perform the activity about which they are interviewed. In some cases, people who had worked a long time as developers, but were currently performing different tasks, provided estimates that differed significantly from the estimates of other experts.

8.3 Application of Elicitation Techniques

One of the most important lessons learned was that the way in which questions are posed could influence the estimate. One question addressed the practical maximum effort for correcting defects. During the first pilot test of the elicitation questions about the different scenarios were asked in the following way: “Under what scenarios would you expect a maximum effort value and what would be that maximum value then?” followed by “What would be then a value for the practical maximum?” This question led the expert to think of “pathological” instances of defect corrections and their abnormally large associated effort. This had a significant impact on the practical maximum estimate. During the de-briefing after pilot testing, the experts concurred with this observation. Therefore, the question in the subsequent interviews was then phrased as “Under what scenario would you expect a practical maximum effort value and what would be that practical maximum value then?” Compared to the experts in the interviews, the estimates of the pilot testers showed significantly higher maximum values. This is a clear indicator that the first way of posing the questions had very likely introduced bias.

A second observation is that people had difficulties to estimate some of the effort values. It was difficult to estimate the correction effort for analysis or design defects detected in late testing phases. This could be attributed to the fact that the tasks for correcting these defects are not as easy to visualize and remember and are not performed as often as, for example, the correction of code defects. Thus, better support for these kinds of questions should be sought for. It was also difficult to estimate the effort to correct one defect after inspections. Some experts had difficulty including the fixed effort occurring once per inspection in their estimates (e.g., checking the document into the configuration system, performing administrative tasks). Thus, in future the model should be refined in the sense that the effort spent on correcting defects and the effort spent on administrative tasks should be elicited separately.

A third observation is that the responses of the experts can significantly differ because of their varying experience. In Section 6.4 we discussed that an expert who is responsible for a very complex part of the system can provide estimates that are significantly different from other experts who are responsible for less complex system parts. Similarly an expert who is currently not performing the task about which he is to provide an estimate can provide different estimates than an expert who is performing the task. Therefore it is necessary to screen all estimates and explain differences in the estimates. We must identify the most plausible causes for the differences and decide, for example, whether to include or exclude an estimate in/from the simulation procedure.

9 Conclusions

The contribution of this paper is four-fold. First, a cost-effectiveness model was proposed by generalizing the assumptions of the cost-effectiveness model proposed by Kusumoto. Second, based on the existing research on expert knowledge elicitation and simulation, a method was provided to estimate the cost-effectiveness of inspections by combining project measurement data and expert opinion. The main objective of these two contributions was to provide a rigorous yet practical approach to the evaluation of inspections from an economic standpoint. It is important as inspections are a widely varying but widespread technique in software development organizations. Several authors have noted the difficulty of making them systematic and convincing developers they are useful. It is important to note that many elements in our approach can be easily tailored to the evaluation of technologies other than inspections. Third, a case study was performed to assess the feasibility of the method and the plausibility of the results. Finally, lessons learned regarding the development and application of the model were also drawn to help future applications.

In particular, three outcomes from the case study are worth highlighting:

1. The quantitative results provide further evidence that inspections can be very beneficial, especially on early developed artifacts.
2. However, it is not clear that inspections are always beneficial, under all circumstances, like it is sometimes suggested in the literature. In our study, for example, the usefulness of code inspections should be investigated further as substantial gains have not been clearly demonstrated.
3. Just the fact of involving the developers and testers in assessing a technology like inspections showed to be beneficial to the quality management team. Interviewees felt they contributed to the evaluation and betterment of current development practices, as opposed to being imposed changes.

10 References

[1] A. F Ackerman., L. S. Buchwald, and F. H. Lewsky. Software Inspections: An Effective Verification Process, IEEE Software, vol 6, no. 3, pp 31-36, 1989.

[2] L. Briand, K. El Emam, O. Laitenberger, and T. Fußbroich, Using Simulation to Build Inspection Efficiency Benchmarks for Development Projects, in Proceedings of the 20th International Conference on Software Engineering, pp. 340-349, 1998.

[3] L. Briand, K. El Emam, and F. Bomarius, COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment, in Proceedings of the 20th International Conference on Software Engineering, pp. 390-399, 1998b.

[4] L. Briand, B. Freimut, and F. Vollei, Assessing the Cost-Effectiveness of Inspections by Combining Project Data and Expert Opinion, Proceedings of the 11th International Symposium on Software Reliability Engineering, pp. 124--135, 2000.

[5] J. Collofello and S. Woodfield, Evaluating the Effectiveness of Reliability-Assurance Techniques, Journal of Systems and Soft-ware, vol 9, no. 3, pp. 191-195, 1989

[6] R. G. Ebenau and S. H. Strauss, Software Inspection Process., McGraw Hill, 1993.

[7] N. Fenton and S. Pfleeger, Software Metrics: A Rigorous and Practical Approach, Thomson Computer Press, 1996

[8] L. A. Franz and J. C. Shih, Estimating the Value of Inspections and Early Testing for Software Projects, CS-TR- 6, Hewlett--Packard Journal, Dec. 1994

[9] T. Gilb and D. Graham, Software Inspection. Addison-Wesley Publishing Company, 1993.

[10] R. Grady, Practical Software Metrics for Project Management and Process Improvement, Prentice-Hall, 1992.

[11] R. Grady and T. van Slack, Key Lessons in Achieving Widespread Inspection Use, IEEE Software, vol 11., no 4, pp. 46-57, 1994
[12] E. Hofer, On surveys of expert opinion, Nuclear Engineering and Design, vol. 93, no. 2-3, pp. 153-160, 1986.
[13] S.C. Hora and R.L. Iman, Expert opinion in risk analysis: the NUREG-1150 methodology, Nuclear Science and Engineering, vol. 102, pp. 323-331, Aug. 1989.
[14] M. Höst and C. Wohlin, A Subjective Effort Estimation Experiment, International Journal of Information and Software Technology, Vol. 39, No. 11, pp. 755-762, 1997.
[15] P. Jalote and M. Haragopal, Overcoming the NAH Syndrome for Inspection Deployment, in Proceedings of the 20th International Conference on Software Engineering, pp. 371-78, 1998.
[16] C. Jones, Software Quality, Thompson Computer Press, 1997.
[17] M. Jorgensen, A Review of Studies on Expert Estimation of Software Development Effort, Journal of Systems and Software, vol. 70, pp. 37-60, 2004.
[18] D. Kahneman, P. Slovic, and A. Tversky, eds., Judgement under uncertainty: Heuristics and biases. Cambridge University Press, 1982.
[19] D.H. Kitson and S.M. Masters, An analysis of SEI software process assessment results: 1987-1991, in Proceedings of the 15th International Conference on Software Engineering, pp. 68-77, 1993.
[20] S. Kusumoto, K. Matsumoto, T. Kikuno, and K. Torii, A new metric for cost-effectiveness of software reviews, IEICE Transactions on Information and Systems, vol. E75-D, no. 5, pp. 674-680, 1992.
[21] O. Laitenberger, S. Vegas, and M. Ciolkowski. The State of the Practice of Review and Inspection Technologies in Germany. Technical Report ViSEK/011/E, Virtuelles Software Engineering Kompetenzzentrum (ViSEK), 2002.
[22] M. A. Meyer and J. M. Booker, Eliciting and Analyzing Expert Judgment: A Practical Guide, Academic Press, Ltd., 1991.
[23] A. Mosleh, V.M. Bier, and G. Apostolakis, The elicitation and use of expert opinion in risk assessment: a critical review, in Probabilistic Safety Assessment and Risk Management: PSA '87, vol. 1 of 3, pp. 152--158, 1987.
[24] A.N. Oppenheim, Questionnaire Design, Interviewing and Attitude Measurement. Pinter Publishers, 1992.
[25] Palisade Corporation, 31 Decker Road; Newfield, NY USA 14867, @Risk - Advanced Risk Analysis for Spreadsheets.
[26] R. A. Radice, High Quality Low Cost Software Inspections, Paradoxixon Publishing, 2002.
[27] D.F. Rico, ROI of Software Process Improvement, J. Ross Publishing, 2004.
[28] T. Rosqvist, M. Koskela, and H. Harju, Software Quality Evaluation based on Expert Judgement, Software Quality Journal, vol. 11, pp. 39-55, 2003.
[29] T. Saaty, The Analytic Hierarchy Process, McGraw-Hill, 1990.
[30] G. Sabaliauskaite, Investigating Defect Detection in Object-Oriented Design and Cost-Effectiveness of Software Inspection, PhD Thesis, 2004.
[31] R. van Solingen. Measuring the ROI of Software Process Improvement. IEEE Software, pp. :32-38, May 2004.
[32] D. Vose, Quantitative Risk Analysis: A Guide to Monte Carlo Simulation Modeling. John Wiley Sons, 1996