Sample-Driven Inspections

Thomas Thelin and Håkan Petersson
Dept. of Communication Systems,
Lund University
{thomas.thelin, hakan.petersson}@telecom.lth.se

Claes Wohlin
Dept. of Software Eng. and Computer Science
Blekinge Institute of Technology
claes.wohlin@bth.se

Abstract

The main objective of software inspections is to find faults in software artefacts. The benefits of inspections are reported from researchers as well as software organizations. In some studies, the fault detection in inspections has shown to be more efficient than other validation and verification activities. A problem, however, is that inspections sometimes are not as efficient and effective as expected. The reason may be that the software artefact inspected contains few faults. In addition, when a software project runs late, inspections are often not properly conducted. This leads to that many faults are not detected, valuable time is lost and people's trust in inspections is affected negatively. Sample-Driven Inspections (SDI) provides a solution to these problems. The concept of SDI uses sampling, inspection and resource scheduling to increase the efficiency of an inspection session. SDI uses a pre-inspection phase in order to determine which artefacts need more inspection time, i.e. which artefacts contain most faults. The second phase of SDI is a main inspection with a special attention on the artefacts with most faults. In this paper, the SDI method is described and empirical evidence is provided, which indicates that the method is appropriate to use. A Monte Carlo simulation is used to evaluate the proposed method. Furthermore, the paper discusses the results and important future research in the area of SDI.

Keywords

Empirical study, Monte Carlo Simulation, Software Inspection.

1. Introduction

High quality, and by that reliability, is built into the software throughout its development. Thus, techniques to increase quality throughout software development are important to ensure delivery of high quality software. Inspection is an efficient technique to detect faults throughout the software development process. Several research papers have reported on the benefits of using inspections [4][7][8][15]. The main purpose of inspections is to detect faults. Empirical research focuses on improving inspections to be more effective and efficient. Different inspection processes, and variants of them, have been proposed since the formalization of the first inspection process [7]. In this paper, inspections are considered to have three main phases: preparation, meeting and fault correction. The aim of these phases is fault searching, fault gathering, and fault correction, respectively.

Different aspects of making software inspections more efficient and effective have been investigated empirically by many researchers. The research focuses on three improvement factors for inspections, *lead-time*, *effectiveness* and *efficiency (effort)*. Lead-time reduction has been evaluated by cancelling the inspection meeting and is defined as the calendar time between two points during development. Efficiency and effectiveness have been investigated by improving the reading technique and by changing the process. Efficiency is defined as the number of (severe) faults found per time unit and effectiveness is defined as the number of (severe) faults found of the total number of (severe) faults.

Sample-Driven Inspections (SDI) is a method to reduce the effort in an inspection session. The aim of the method is to concentrate the inspection effort on the software artefacts that contain most faults. SDI divides the fault searching into two parts. First, samples of the artefacts are inspected in order to estimate which artefacts contain most faults (pre-inspection). Second, an inspection is carried out for these artefacts (main inspection). It is important to notice that SDI does require neither any special type of reading technique nor any special preparations for the reviewers. This means that any type of reading technique can be used when applying SDI.

Gilb and Graham [8] and Burr and Owen [2] have proposed sampling of software artefacts, although leaving several open research questions, including, for example, sample size and number of reviewers. They suggest inspecting part of a software artefact in order to determine whether the artefact is ready for the main inspection. Gilb and Graham [8] argue that the same types of faults exist in different

forms throughout the same document. Hence, a sample of one or some pages would be enough to get a picture of the fault distribution in a software document. Gilb and Graham, as well as Burr and Owen, describe sampling as a way to make inspections more efficient and effective. SDI uses the sampling part and defines a method, which provides software organizations with the opportunity to increase their inspection performance.

This paper consists of two parts. The first part introduces the concept of SDI, which constitutes three parts: sampling, inspection, and resource scheduling. Neither of these parts is new. The novelty is the combination and empirical investigation of the method. The second part is a Monte Carlo simulation that provides an investigation of SDI as an inspection method. The important parameters that are investigated are sample accuracy, the number of reviewers, the ability of the reviewers and the percentage inspected in the pre-inspection (sample size).

The main result from the simulation is that the method works and is appropriate to use. The results are valid even if the sample accuracy is low, which indicates that the method is robust enough to be further studied. Further work includes experimentation with sampling and effort scheduling, as well as case studies in industrial settings.

The paper is structured as follows. The Sample-Driven Inspection method and different aspects related to SDI are presented in Section 2. In Section 3, the simulation model is described together with a discussion of the document profiles used, where a document profile refers to how the faults are distributed across the document. The results and a discussion of the results are presented in Section 4 and Section 5. In Section 6, the main conclusions are presented and some areas for future research are identified.

2. Sample-Driven Inspections: Method Description

2.1. Overview and Motivation

In software development, driven by market forces, the projects have to move very fast through the development phases because of the strong demand of releasing new or upgraded products as often as possible. This is the general situation in software development today. In order to deliver software with appropriate quality, different methods for quality control and improvement are used. One such method is software inspections [7].

Inspections are an integrated part of a company's development process. The project manager, or someone with similar responsibilities, has to take care of the planning of the inspections and consider a number of different questions: *how* to inspect, *what* to inspect, *when* to inspect, and *who* should perform the inspection. Since inspections require resources that could otherwise be used for development, the concept of resource scheduling is closely related to performing inspections.

Unfortunately, project progress is still in many cases measured with measures like lines of code or pages written. Since inspections is a quality assurance activity, and not a producing activity, it achieves a low score with such measures. Low scoring of an inspection often leads to that inspections is the first activity to be cut down when projects start to run out of time. The project manager finds himself/herself in the position of having to choose parts of the product to inspect since there is no time to inspect all. If accepting that this situation occurs, the best solution would be to schedule the available inspection resources to the software artefacts that need it the most. However, since the quality of the documents is unknown, it has to be estimated.

The estimation can be achieved by applying sampling to support the scheduling task. In addition to the *inspection* and *resource scheduling* tasks, we add the task of *sampling*, see Figure 1. By inspecting a small sample of all available documents, it is possible to estimate which artefacts have the lowest quality and make sure that these get most attention during the inspections.

The concept of sampling documents to support inspections has been briefly mentioned in a couple of sources, for example, [2][8][10]. However, the impact and magnitude of the possible effort gain have not been investigated. In this paper, we present and make an initial investigation of the concept of Sample-Driven Inspections (SDI).

2.2. Sampling and Resource Scheduling

Each of the three parts of SDI, see Figure 1, allows for a lot of variation. The inspections can be performed with different techniques, the sampling can be made in numerous ways and the resource decisions, based on the sampling, can be done in various ways as well.

The sampling part is of special focus in SDI. It is crucial to whether the method will work properly or not. The goal of the sampling is to achieve, with a minimum of resources, a representative sample of each document's quality. This would lead to a correct evaluation of which document that needs most attention during an inspection. Different sampling strategies are probably needed for different types of software documents. A textual document, for example, a requirements specification or a user guide can be sampled by choosing the number of pages to cover beforehand. While for code or design documents it might be easier to first perform a minor inspection during limited time and finally estimate how large part that was covered.

After having decided upon a sampling technique, the documents are sampled and the samples are inspected.

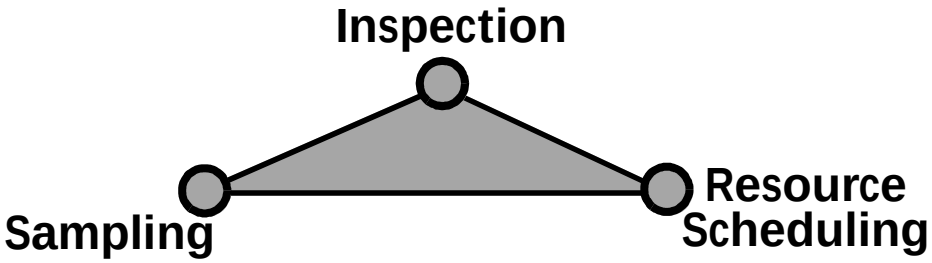

Figure 1. General view of Sample-Driven Inspections.

Based on the outcome, quality estimations of the documents are made. This *pre-inspection* can be conducted by either one reviewer or a group of reviewers. After the pre-inspection, the documents are sorted and resources are distributed.

In many cases, not only the quality of the document determines how important it is to inspect. Different parts of the product could be more safety critical or contain functionality of higher importance than other parts. The prioritization of where to put the inspection effort will then be based on a combination of quality and importance. Since the criticality and importance of documents vary between companies as well as within a company, the method of prioritizing has to be designed for the specific case.

The resources can also be divided in many ways. An example is that different number of reviewers inspects different documents. Another example would be to assign a given amount of inspection time to each document. This is an important subject for further studies of SDI.

2.3. Investigation

It is obvious that the SDI approach will aid the scheduling of resources if representative sampling is possible. Within the mere definition of *representative* lies the fact that it gives a good enough picture of the quality of the documents. However, even if representative sampling is assumed, it is of interest to investigate to what extent the method works. In this paper, the method's magnitude of gain is first investigated making the assumption of representativeness and second, the effect of relaxing this assumption is investigated. This is evaluated by running a process simulation [9], of an inspection process. The model used for simulating software inspections is closely related to, and based on, the model used for capture-recapture simulation, see for example [3][6][13].

To investigate the possible effects, we create a conceptual model of how SDI could be implemented within a software organization, see Figure 2. The simulation of the model is used to evaluate different parameter settings. The available documents are sampled, and based on a pre-inspection of these samples the resources are scheduled. The scheduling is based on the sorting of the documents, which is based on the perceived fault content from the pre-inspection. The main inspection is then performed with the assigned resources.

The model in Figure 2 is very general and some restrictions have to be made in order to make an investigation possible.

1. Faults used in the sorting of the documents are of equal importance – This can be viewed either as if all faults found are used or only the severe faults are used when sorting the documents.
2. Documents are of equal importance – This assumption can be viewed from a project manager perspective. He/she could select a number of equally important documents to apply the SDI to.
3. Representative sampling of a document is possible – This assumption is our starting point. However, investigations of SDI's behaviour, when disturbing the representativeness, is performed and further discussed in Section 2.5.

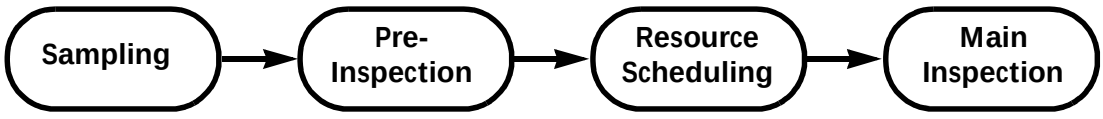

Figure 2. Conceptual model of SDI in a company.

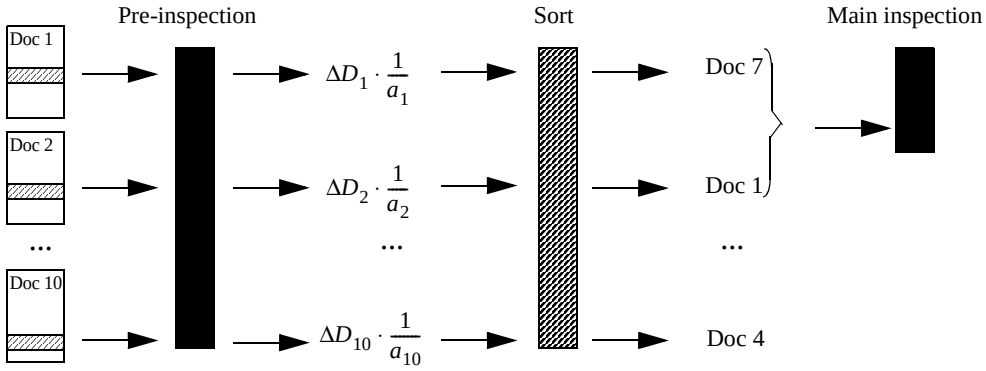

Figure 3. The steps in the Sample-Driven Inspection method.

The simulation procedure of SDI can briefly be described as follows. First, a sample of each of the available documents is inspected. Second, the documents with the highest estimate of faults are selected and the main inspection is concentrated on these. The following four steps gives a more detailed explanation, see also Figure 3:

1. Inspect a sample, a_i, of each software artefact i, where a_i is a percentage figure of the document. Note the number of faults found, ΔD_i.
2. Estimate the total number of faults by multiplying the number of faults found in each artefact with $1/a_i$.
3. Sort the artefacts in descending order according to $\Delta D_i \cdot \frac{1}{a_i}$.
4. Focus the inspection effort on the artefacts with highest rating.

Note that the scaling of the found faults is only necessary if the samples are of different sizes.

2.4. An Example Scenario

As an example, assume that five design documents are to be inspected. The project manager selects five documents of equal importance and decides that 20% of the documents are to be inspected. Since the documents are about 10 pages, he/she selects two of the pages that he/she thinks is appropriate as a sample of the documents. Prior to the pre-inspection, the project manager has decided that only severe faults should be considered when sorting the documents.

The five documents contain 5, 5, 10, 15 and 20 severe faults. (Of course, this is not known beforehand.) In the 20% pre-inspection 0, 1, 2, 1 and 3 severe faults are found. This leads to a sorting of the document as (20, 10, 5, 15, 5). The project manager decides that the focus in the main inspection will be put on the documents where 3 and 2 faults were found. If we count the approximate number of faults that have been exposed to an inspection, we find (0.20*(5+5+10+15+20)+0.80*(10+20)) = 35 faults. Note that *exposed* includes all faults that exists in the reviewed material not only the ones that the reviewer found.

This means that it is potentially possible to detect about 64% of the faults using 52% of the effort, compared to inspecting every document fully or selecting a document to inspect by random. This example shows that important project resources and time could be saved by the use of SDI.

2.5. Document Profiles

In Section 2.3, the assumption that representative samples are possible to achieve is made. In the simulation, this is done by randomly distributing the faults in the documents as well as randomly selecting what to pre-inspect. To investigate the effect of what will happen when disturbing the representativeness, both the random distribution and the random selection are removed.

If a document is divided into 10 equally sized parts and define d_i as the fraction of the total number of faults in part *i*. Then an approximate measure of how equally distributed the faults are in the document would be to calculate the standard deviation of $\{d_i, i=1...10\}$. We denote this measure S. Even if faults are randomly distributed, the most likely value of S is not zero. If the number of faults were infinite, this would however be the case. For example, with 100 faults randomly distributed, S has an average of 0.031.

To disturb the effectiveness in sampling, we force the documents to have their faults distributed according to specific profiles. The profile assigned to a document is randomly chosen. Five profiles, a to e, are used, see Figure 4. The S measure of these profiles are {0.033 0.062 0.100 0.141 0.211}. The mix of the different profiles is 10:5:5:5:1 where with *a* is the most common and *e* is the most unlikely.

The design and mixture of these profiles do not rely on any empirical information of how faults are distributed in

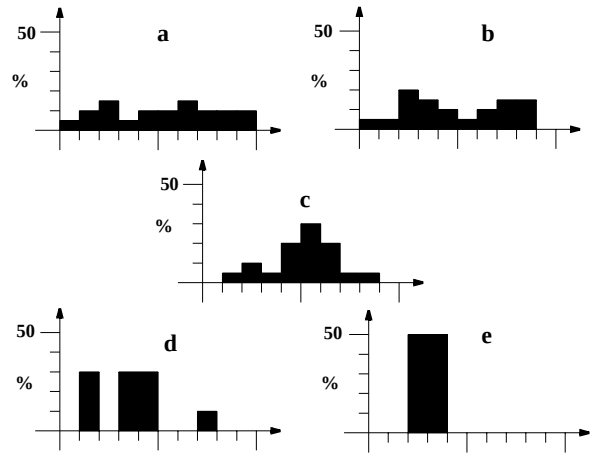

Figure 4. Fault distribution profiles

documents. The approach and design are made to study the effect of disturbing of the sampling part of SDI. An empirical investigation of how faults are distributed in reality would be of interest, especially when designing the sampling technique to use.

The robustness of SDI using these profiles is discussed further in Section 4.1.

2.6. Research Questions

Under the assumption that it is possible to achieve representative samples, it is obvious that the SDI approach would help the resource planning. But to what extent? Are the positive effects large enough to justify research of how to identify a representative sample? This study concentrates on investigating whether the method succeeds and how it behaves when changing different parameters. The SDI method is evaluated in a Monte Carlo simulation. The goal of the simulation is to answer the following questions:

- How large part of the artefact has to be pre-inspected in order to make a good enough sorting?
- How does the number of reviewers affect the result?
- How do the abilities of the reviewers affect the result?
- How does the representativeness of the samples affect the result?

3. Simulation Model of Sample-Based Inspections

A Monte Carlo simulation is used to investigate whether the SDI method gives reliable results. Furthermore, the simulation is designed to evaluate the research questions stated in Section 2. This section describes the simulation model and explains the parameters used in the model.

The simulation model consists of three parts. The first part is designed to simulate one inspection (Section 3.1). The second part is designed to simulate SDI (Section 3.2). The third part is used to evaluate the SDI method (Section 3.3). These parts are described in the subsequent sections.

The simulation is based on a model called M_{th} in capture-recapture [11]. The model assumes that reviewers as well as faults are independent. This choice of model was decided upon after observing empirical data, showing the M_{th} model to be the most realistic one. A consequence of using this model is that if one reviewer has higher probability to detect a specific fault than other reviewers, he/she has higher detection probability of all faults. Another model that does not have this constraint was under consideration. However, during the model validation, this less constraint model showed to be less realistic using empirical data of the effectiveness.

3.1. Simulation Model of One Inspection

To simulate one inspection five parameters have to be considered. The five parameters are:

- *Size of a document* – A document consists of 1000 places, where one fault can be injected in one place. The faults are randomly injected at these places, using a uniform distribution. This means that the faults can be injected in any place with the same probability.
- *Number of Reviewers* – The number of reviewers for one inspection is part of the simulation model of SDI and is described in Section 3.2.
- *Ability of reviewers* – The abilities of the reviewers represent the reviewers' ability to find faults and perform software inspections. The abilities of the reviewers are designed to investigate the performance of SDI. Therefore, the choices of these abilities are described in Section 3.2. The ability of reviewer j is denoted r_j.
- *Number of faults* – The number of faults in a document is part of the evaluation model and is described in Section 3.3.
- *Probability of faults* – The probabilities of the faults are assumed to follow a uniform distribution. Three classes of faults are used: easy, moderate and difficult faults. The use of a uniform distribution means that the number of easy, moderate and difficult faults is equally probable in a document. The detection probability for fault i is denoted f_i.

Note that the probability to find a specific fault for a specific reviewer is determined by multiplying the ability of the reviewer with the detection probability of the fault ($F_{ij}=f_i \cdot r_j$).

Since only a percentage of a document is inspected in the pre-inspection, a combination of where to look (the sam-

pling) and where the faults are located is the key to whether the reviewers find the faults. The sampling part is further evaluated using the profiles discussed in Section 2.5.

3.2. Simulation Model of SDI

The design of the simulation model of SDI is based on empirical data collected from 30 experiments and case studies in the area of software inspections [7] and capture-recapture [5][11]. An overview of the simulation model of SDI is shown in Table 1 and Figure 5. Both the pre-inspection and the main inspection parts are simulated.

The simulation model of SDI is designed to study three parameters:

- *Number of reviewers* – The number of reviewers is selected to be one, three or five. The same number of reviewers is used for pre-inspection and main inspection.
- *Percentage of pre-inspection* – To achieve accurate results, five percentages are used to investigate the size of the fractions in the pre-inspection (10%, 20%, 30%, 40% and 50%). A linear relation is assumed between the sample size and the effort needed to perform the pre-inspection and the main inspection. Since it is assumed that the resources are limited, the percentage values decide the number of documents to be inspected during the main inspection. For example, if 10% of a document is pre-inspected, more documents can be inspected during the main inspection. On the other hand, if 50% of all documents are inspected, a more confident decision can be made of which documents that are fault-prone. However, fewer documents are inspected during the main inspection. In other words, the decision becomes a trade-off between the confidence in pre-inspection and the resources left to conduct the main inspection.
- *Ability of reviewers* – The abilities of the reviewers are divided into three categories, see Table 1. The categories investigated are when all reviewers has a high ability to find faults (All good), all reviewers has a low ability to find faults (All bad) and a mixture of the abilities of the different reviewers (Mixed).

The choice of the abilities of the reviewers was made through an investigation of 30 data sets from empirical studies. The mean of the F_{ij} values of these data sets was found to be 0.27. In order to resemble this characteristic, the mean of the abilities in the mixed case is set to 0.6. Since the probabilities of the faults are from the uniform distribution[1], this leads to the mean of the generated data sets in this case is 0.6 * 0.5 = 0.3. For the other cases, lower respectively higher mean values are chosen. The data sets were collected from different types of software inspections, for example, perspective based reading [1], checklist based reading [4], ad hoc inspections, and on different types of software artefacts, for example, requirements and code documents. The data sets are further described in [14]

The simulation varies three parameters: the number of reviewers, the percentages for the pre-inspection and the ability of the reviewers. This leads to 45 combinations. For each of these combinations, 1000 inspections are simulated in order to get enough evaluation data. However, the same documents are used for the 45 cases.

3.3. Evaluation Model of SDI

The evaluation model of SDI consists of three parameters, *the number of documents*, *the number of faults* in the documents and a resource limit, which in this paper is called *work points*. Furthermore, to evaluate the success of the method *the number of exposed faults* and *the number of found faults* are measured.

After a pre-inspection has been conducted, the documents are sorted in descending order in terms of estimated faults in the document, see Section 2. To evaluate whether the method sorts the documents in correct order, 10 documents are used, which contain 10 to 100 faults in steps of 10, i.e. a total of 550 faults, see Figure 5.

The optimal result would be if the documents were sorted in correct order. Hence, the documents containing most faults can be selected for the main inspection. The success of the method is, however, not dependent of an exactly correct ranking. It is sufficient if the artefacts are sorted so that the correct artefacts are pinpointed for the main inspection

1. The uniform distribution gives a mean of r_i at 0.5.

Table 1: The three different cases simulated during the study.

	Ability		
	1 Reviewer	**3 Reviewers**	**5 Reviewers**
All Good	0.9	0.8, 0.9, 1	0.8, 0.85, 0.9, 0.95, 1
Mixed	0.6	0.4, 0.6, 0.8	0.3, 0.5, 0.6, 0.7, 0.9
All Bad	0.3	0.2, 0.3, 0.4	0.2, 0.25, 0.3, 0.35, 0.4

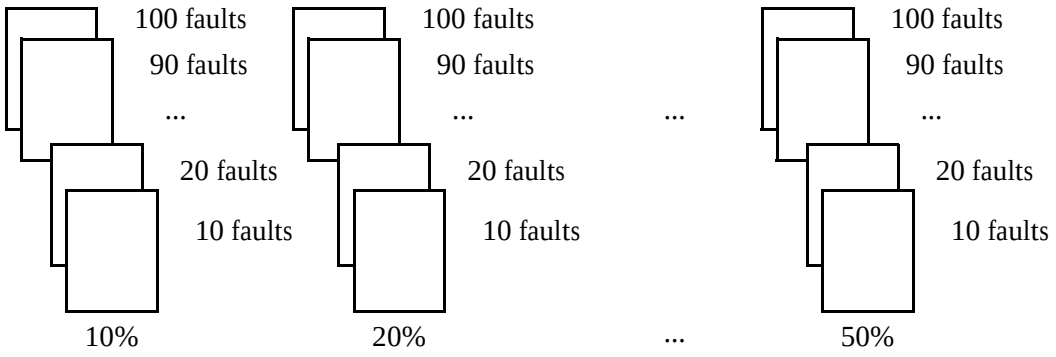

Figure 5. An overview of the simulation model. The percentages are varied from 10% to 50%. The faults are varied from 10 to 100 faults.

To compare the parameters investigated, work points are used as a means for available resources. The different work points used are:

- 100 points – the resources used for inspecting one whole document.
- 550 points – the available resources in one inspection session.

Ten documents are used in one inspection session. This means that 5.5 documents of 10 can be inspected. If 10% is used as a sample, 10% of 5 documents are inspected plus, hopefully, the 5 most fault-prone. In the 50% case, 50% of 9 documents are inspected plus one whole document. If two or more documents are predicted to be equally fault-prone, the remaining work points are distributed equally among these.

The same amount of work points is used for one, three and five reviewers.

The evaluation is carried out by measuring the faults that could be found (exposed) and the faults that are actually found (found) in a main inspection.

- **Exposed Faults** – This is a measure of the sorting of documents. The measure counts the faults that are exposed in the pre-inspection and the main inspection. In the pre-inspection, a sample of all documents are used and in the main inspection, only the documents that are predicted to contain most faults are used.
- **Found Faults** – This is a measure of the whole procedure and considers both the pre-inspection sampling and the main inspection. The measure counts the faults that are actually found during the pre-inspection and the main inspection.
- **Number of Times Selected** – This is a measure for the fault-prone artefact evaluation and consider the number of times the two documents with the highest number of faults are selected for the main inspection.

4. Empirical Evaluation of Sample-Based Inspections

In this section, the results and observations are presented. The results are further discussed in Section 5. This section is divided into two subsections. The first subsection discusses the results after the pre-inspection and the second subsection discusses the results after the main inspection.

4.1. Pre-inspection

In Figure 6 and Figure 7, boxplots of the results of the pre-inspection, when assuming that the samples are representative, are shown. The results of having one and three reviewers in the pre-inspection phase are presented. The results for five reviewers are similar. The mean number of exposed faults when choosing documents by random is 302.5. This value is shown as a straight line in the boxplots.

A number of observations can be made from the boxplots:

- The higher percentage pre-inspected, the less dispersion. This reflects the fact that a higher percentage leads to a more confident classification. However, fewer resources are left for the main inspection.
- The more reviewers used the more reliable results. However, the results show that the differences are very small. This points in the direction that few reviewers are needed in the pre-inspection.
- The higher ability of the reviewers, the less dispersion is obtained. This is a consequence of that competent reviewers finds more faults. Hence, the sorting of the documents becomes more reliable.
- For the mixed case with a pre-inspection of 20%, in median 68% of the faults are exposed. This means that using 55% of the effort, 68% of the faults can be found

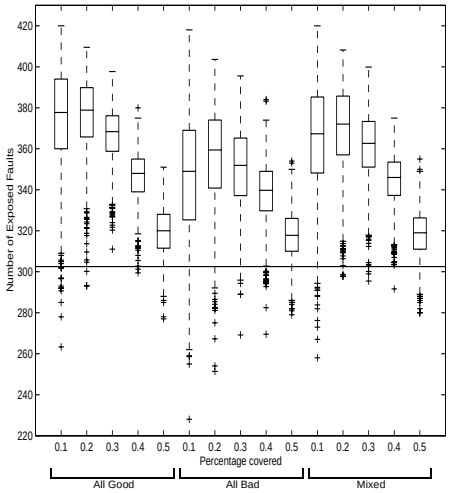

Figure 6. Number of exposed faults for one reviewer.

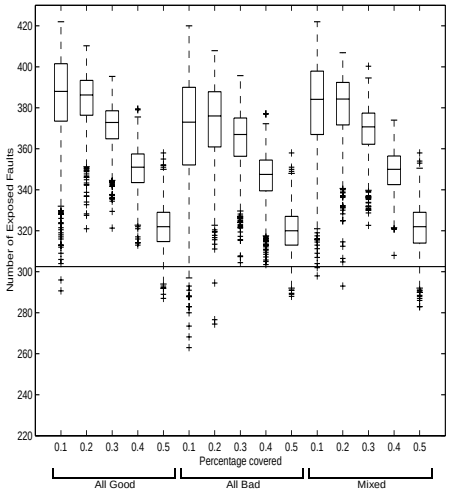

Figure 7. Number of exposed faults for three reviewers.

during the inspection. Consequently, SDI exposes more faults per resource used.

In Figure 8 and Figure 9, boxplots of the evaluation of sorting using the profiles are shown, see Section 2.5.

The results do not become this good when using document profiles. However, the method is robust to deviations from the assumption of representative samples. For the mixed case, pre-inspecting 20%, 63% of the faults are exposed (using 55% of the effort). For some cases the reduction of exposed faults are larger. However, in neither of the cases the median of the sampling approach is lower than the median value when not using sampling.

The simulation of the profiles shows that sampling is important. Although SDI provides smaller profit when the samples are less representative, it is still better compared to selecting the documents by random.

4.2. Main Inspection

The effect of the abilities of the reviewers is not large when investigating the number of exposed faults. This is a consequence of the robustness of SDI, i.e. as long as the reviewers have equal abilities for all documents, the order of

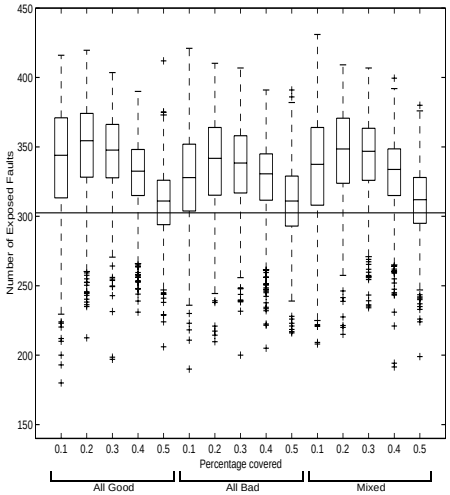

Figure 8. Number of exposed faults for one reviewer when the documents use the profiles discussed in Section 2.5.

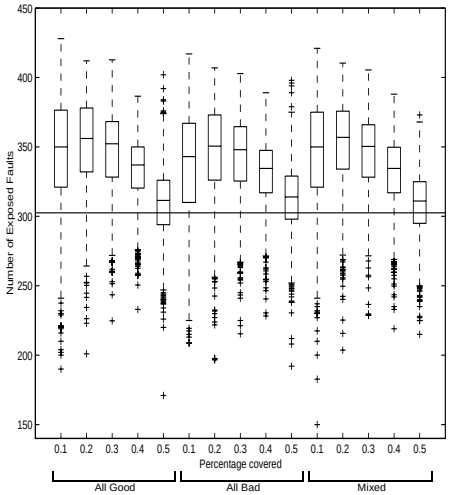

Figure 9. Number of exposed faults for three reviewers when the documents use the profiles discussed in Section 2.5.

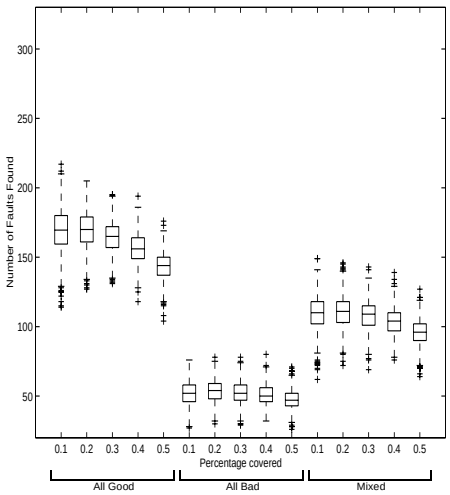

Figure 10. Number of found faults for one reviewer.

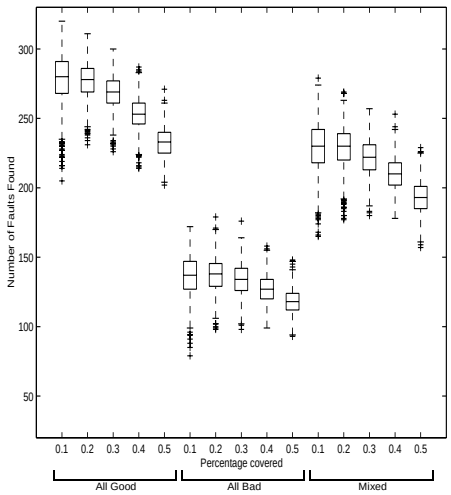

Figure 11. Number of found faults for three reviewers.

the sorting will not be affected.

However, the main inspection will be affected by the abilities of the reviewer. This effect is shown in Figure 10 and Figure 11. In these figures, the total number of faults found from both the pre-inspection and the main inspection is shown.

Hence, the ability of the reviewers is very important for the success of an inspection. However, the ability does not affect the pre-inspection part of SDI much.

The case when a small number of artefacts contains substantially more faults than the others has also been studied. As expected, SDI performs very well. In summary, SDI performs very well when the differences in quality between the documents are large. If the differences are small, it is more difficult to sort the documents in a correct order. On the other hand, it is also much less critical. Thus, SDI works well when it is most needed.

5. Discussion

5.1. Research questions

In Section 2, four research questions were posed in this study.

- How large part of the artefact has to be pre-inspected in order to make a good enough sorting?
- How does the number of reviewers affect the result?
- How do the abilities of the reviewers affect the result?
- How does the representativeness of the samples affect the result?

As shown in Section 4 almost all values lies above the mean of what would be the case if the 550 work points were spent randomly among the documents. This leads to the conclusion that the SDI method does work.

The largest variation in the results comes from the choice of the sample size. All of the investigated sample sizes give better results than the random approach. Even inspecting only 10% gives a fairly good picture of the number of faults in each artefact and is better than choosing documents randomly. The choice of the size of the sample is also a matter of trade-off between mean and dispersion. Pre-inspecting 50% results in a very small dispersion but not much improvement in mean compared to the 10% case.

Regarding question two and three, neither the number of reviewers nor their abilities affect the outcome in terms of number of exposed faults very much. The dispersion in the three-reviewer case is smaller and the median is somewhat better compared to the one-reviewer case. The same can be seen when comparing the *all good* reviewer case with the *all bad* reviewer case. This is expected since the risk of missing a fault is smaller the more and better reviewers that participate in the pre-inspection.

The last question is more difficult to answer. It is obvious when disturbing the sampling that the yield from SDI becomes smaller. Even if, Section 4.1 shows concrete measurable results it is not clear how well the used disturbance reflects *true* circumstances. In a real-world inspection, the yield of SDI will be a tussle between how the faults are distributed and how well the sampling technique works. However, with the circumstances used in our simulation, SDI still delivers better results than merely selecting the documents by random.

5.2. Applying SDI

One aspect contributing to the robustness of SDI is that it is not important that the scaling of the found faults lies exactly at the true value, neither is it important that the sorting becomes exactly correct. The only important issue is that the correct artefacts are selected for the main inspection. For example, in the 10% case, five documents are chosen for the main inspection, which means that it is only a matter of dividing the documents into two halves.

Even if the number of reviewers and their abilities do not affect the scores of exposed faults very much, it does affect the actual outcome of the inspection, as can be seen in Section 4.2. When it comes to actually finding the faults, it is very important how many and how good the reviewers are. It must be remembered that SDI does not improve the reviewers' chance of finding faults; it makes the best out of the given situation.

As mentioned earlier, the key to success lies in the sampling. The application of SDI relies on that organizations learn how to select a representative sample. This could be made either through random sampling or through selecting an appropriate part of the artefact, where appropriate means a part that is close to the mean quality of the artefact. A related issue is to study the effect of first deciding on what sample to pre-inspect and then determine the size of that sample. These matters ought to be further investigated in a series of controlled experiments where different approaches of sampling as well as the SDI approach as such can be evaluated. It is likely that the sampling method will be highly dependent on the type of artefact as well as the organization using the method. The actual sampling procedure probably has to be calibrated for the given situation.

An early recommendation, based on the results in this study, is to use one reviewer to pre-inspect 20-30% of the document and then decide which documents to focus on in the main inspection. In the simulation, 20-30% managed to expose in average 70% of the total number faults. This yield is dependent on the distribution of faults among the documents as shown in Section 4.1. A rather common case in software development is the 20-80 rule. This rule means that often 20% of the modules contain 80% of the faults. The Pareto effect has been shown empirically in, for example, [12], although in this study 20% of the modules accounted for 60% of the faults. In our study of particularly fault-prone documents, SDI is very successful in the case when the quality differences are large.

6. Summary

As a way of increasing the quality of software products, inspections are well established. However, as many other quality improving tasks, inspections take time. When projects run late or are close to a deadline, the time to inspect sometimes is decreased or the product is delivered into the next development phase without being properly inspected.

In this study, a method to allocate the inspection resources efficiently is proposed. The method deals with the case of having a number of documents to inspect but not the resources available to cover them all. The main parts of the method are the pre-inspection phase and the main inspection phase. During the pre-inspection, a sample of a number of artefacts is inspected in order to sort them in terms of fault-proneness. Then, during the main inspection, the artefacts predicted to be most fault-prone are inspected. The number of artefacts inspected during the main inspection is determined by the available resources. The method, however, does require neither the limitation of resources to be decided beforehand nor any special technique for the inspection.

The method was investigated empirically with a Monte Carlo simulation. The results can be summarized as follow:

- The method works and gives an advantage over merely choosing software artefacts by random.
- The method is robust, simple to use and gives reliable results.
- The recommendation is to use one reviewer to pre-inspect 20-30% of the software artefacts.
- The method shows even more promising results when it comes to identify fault-prone software artefacts.

To summarize, SDI works best when the differences in the quality between the documents are large and it is not as good as when the differences are smaller. Thus, SDI is best when it is needed the most.

Further work would be to run a controlled experiment in order to evaluate the method without the limitations of the simulation. The focus of such an experiment would be to investigate if the SDI approach works in a real setting as well as investigate different ways of solving the representative sample problem.

Acknowledgements

This work was partly funded by The Swedish National Board for Industrial and Technical Development (NUTEK), grant 1K1P-99-6121.

References

[1] Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S. and Zelkowitz, M. V., "The Empirical Investigation of Perspective-Based Reading", *Empirical Software Engineering: An International Journal*, 1(2):133-164, 1996.

[2] Burr, A. and Owen, M., *Statistical Methods for Software Quality – Using Metrics for Process Improvements*, International Thomson Computer Press, UK,1996.

[3] Chao, A, "Estimating Population Size for Sparse Data in Capture-Recapture Experiments", Biometrics 45, pp. 427-438, 1989.

[4] Ebenau, R.G. and Strauss, S. H., *Software Inspection Process*, McGraw-Hill, New York, 1994.

[5] Eick, S. G., Loader, C. R., Long, M. D., Votta, L. G. and Vander Wiel, S. A., "Estimating Software Fault Content Before Coding", *Proc. of the 14th International Conference on Software Engineering*, pp. 59-65, 1992.

[6] El Emam, K. and Laitenberger, O., "Evaluating Capture-Recapture Models With Two Inspectors, to appear in *IEEE Transactions on Software Engineering*, 2001.

[7] Fagan, M. E. "Design and Code Inspections to Reduce Errors in Program Development", *IBM System Journal*, 15(3):182-211, 1976.

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

[9] Kellner, M. I., Madachy, R. J., Raffo, D., M., "Software Process Simulation Modeling: Why? What? How?", *Journal of Systems and Software*, 46(2/3):91-105, 1999.

[10] Kit, E. *Software Testing in the Real World – Improving the Process*, Addison-Wesley, USA, 1995.

[11] Miller, J., "Estimating the Number of Remaining Defects after Inspection", *Software Testing, Verification and Reliability*, 9(4):167-189, 1999.

[12] Ohlsson, N. Helander, M. and Wohlin, C., "Quality Improvement by Identification of Fault-Prone Modules using Software Design Metrics", *Proc. of the 6th International Conference on Software Quality*, pp. 1-13, 1996.

[13] Otis, D. L., Burnham, K. P., White, G. C. and Anderson, D. R., "Statistical Inference from Capture Data on Closed Animal Populations", *Wildlife Monographs* 62, 1978.

[14] Petersson, H. and Wohlin, C., "Evaluation of using Capture-Recapture Methods in Software Review Data", *Proc. of the Conference on Empirical Assessment & Evaluation in Software Engineering*, 1999.

[15] Weller, E. F., "Lessons from Three Years of Inspection Data" *IEEE Software*, 10(5):38-45, 1993.