Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Systems and Software 72 (2004) 249–264

The Journal of Systems and Software

www.elsevier.com/locate/jss

Capture–recapture in software inspections after 10 years research—theory, evaluation and application

Håkan Petersson [a], Thomas Thelin [a,*], Per Runeson [a], Claes Wohlin [b]

[a] Department of Communication Systems, Lund University, P.O. Box 118, 221 00 Lund, Sweden
[b] Department of Software Engineering and Computer Science, Blekinge Institute of Technology, Sweden

Received 5 October 2002

Abstract

Software inspection is a method to detect faults in the early phases of the software life cycle. In order to estimate the number of faults *not* found, capture–recapture was introduced for software inspections in 1992 to estimate remaining faults after an inspection. Since then, several papers have been written in the area, concerning the basic theory, evaluation of models and application of the method. This paper summarizes the work made in capture–recapture for software inspections during these years. Furthermore, and more importantly, the contribution of the papers are classified as *theory*, *evaluation* or *application*, in order to analyse the performed research as well as to highlight the areas of research that need further work. It is concluded that (1) most of the basic theory is investigated within biostatistics, (2) most software engineering research is performed on evaluation, a majority ending up in recommendation of the Mh–JK model, and (3) there is a need for application experiences. In order to support the application, an inspection process is presented with decision points based on capture–recapture estimates.

1. Introduction

Software inspection (Ebenau and Strauss, 1994; Gilb and Graham, 1993) is a method to detect faults in software artefacts early in the development cycle. It was first described by Fagan (1976), and since then inspections have been established as state of the practice and have evolved to become a mature empirical research area, as it is of reasonably limited scope to research and nevertheless of substantial practical importance. The research has addressed changes to the inspection process, e.g. (Bisant and Lyle, 1989; Knight and Myers, 1993; Martin and Tsai, 1990; Parnas and Weiss, 1985), support to the process, e.g. (Basili et al., 1996; Eick et al., 1992) and empirical studies, e.g. (Porter et al., 1995; Thelin and Runeson, 2000a). The support to the inspection process includes reading techniques (Basili et al., 1996) and the use of capture–recapture techniques to estimate the remaining number of faults after an inspection (Eick et al., 1992). Furthermore, the benefits of conducting software inspections have been studied in industry (Weller, 1993). Reading techniques are applied to the individually performed part of inspections, in order to aid reviewers with more information of how to search for faults. The purpose is to increase the efficiency and effectiveness. Several reading techniques have been proposed: checklist-based reading (Fagan, 1986), defect-based reading (Porter et al., 1995) perspective-based reading (PBR) (Basili et al., 1996), traceability-based reading (Travassos et al., 1999) and usage-based reading (Thelin et al., 2001a).

Capture–recapture (Chao, 1998) is a statistical method that can be applied to software inspections to estimate the fault content of an artefact. It can be applied to all software artefacts that can be inspected, for example, requirements, design and code documents. Capture–recapture was first applied to software inspections by Eick et al. (1992), and since then a number of papers have been published that evaluate and improve capture–recapture for software inspections, see for

[*] Corresponding author. Tel.: +46-46-222-38-63; fax: +46-46-14-58-23.
E-mail addresses: hakan.petersson@telecom.lth.se (H. Petersson), thomas.thelin@telecom.lth.se (T. Thelin), per.runeson@telecom.lth.se (P. Runeson), claes.wohlin@bth.se (C. Wohlin).

example (Petersson, 2002; Thelin, 2002). The method uses the overlap between the sets of faults found by different reviewers to estimate the fault content. It is assumed that the reviewers work independently of each other and therefore the fault searching has to be performed before, and not during, an inspection meeting. The size of the overlap indicates the number of faults left; if the overlap is large, few faults are left to be detected; if the overlap is small, many faults are undetected. Using statistical methods, an estimation value and a confidence interval can be calculated. This information can be used by inspection coordinators and project managers to take informed decisions, which is exemplified in Section 5.

The first known use of capture–recapture was by Laplace (1786), who used it to estimate the population size of France (Pollock, 1991). In biology, capture–recapture is used to estimate the population size of animals, e.g. the number of fish in a lake, and it is also used in medical research (Chao, 1998). In software engineering, capture–recapture methods have been used together with fault seeding (Mills, 1972). The number of faults remaining after testing are estimated based on the overlap between identified seeded faults and non-seeded faults. Capture–recapture methods have also been utilized in other variants of software testing (Stringfellow et al., 2002; Yang and Chao, 1995).

The purpose of this paper is to summarize the capture–recapture research in software inspections during the past ten years. During these years, a number of research papers have been published. By categorizing the papers, an overview of conducted research is given and areas that need further research are identified. The papers have been classified into three main categories *theory*, *evaluation* and *application*. Several papers have considered the theory and evaluation of capture–recapture. Only one published paper has reported experience from a trial application of capture–recapture in an industrial environment.

To find the relevant literature for the survey, a literature search was carried out. This was made through searching the databases INSPEC [1], IEEE on-line [2], Science Direct [3] and ACM [4] using the keywords "capture recapture", "defect content estimation" and "fault content estimation". In addition, some papers were obtained by personal communications with researchers, known to have published research in the field. Finally, all references in the included papers were checked to guarantee that no referenced paper was missed.

As the application part was found to be weak, an inspection process is presented that uses the capture–recapture estimates as decision points. The process is intended as a guide for practical application of capture–recapture in software organizations.

This paper is outlined as follows. In Section 2, the theory of capture–recapture is presented. A summary of the research in the capture–recapture papers is provided in Section 3 and in Section 4, the generalized findings of the research are summarized. Section 5 presents an inspection process which is integrated with capture–recapture estimates. In Section 6, future research is suggested and in Section 7 a summary of this paper is provided.

2. Capture–recapture

There are many variants of models and estimators based on the capture–recapture principles. An estimator is a formula used to predict the number of faults remaining in an artefact. A model is the umbrella term for a number of estimators based on the same prerequisites. Four basic capture–recapture models are used for software inspections (see Table 1). However, more models have been developed for other domains, but have not yet been investigated for software inspections.

The overlap among the faults that the reviewers find is used as a basis for the estimation. The smaller overlap among the reviewers, the more faults are assumed to remain, and the larger overlap, the fewer faults are assumed to remain. The extreme cases are the following: either, all reviewers have found exactly the same faults, which means that there are probably very few faults left, or none of the reviewers has found a fault that another reviewer has found, which indicates that there are probably many faults left. To estimate the number of faults left, statistical estimators are used, which are designed to model different variations in software inspections.

There are two main categories of models, models for open populations and models for closed populations. Open populations means populations which change in size through births and deaths, while closed populations remain of the same size throughout the counting period. In software inspections, the models for closed populations are primarily applicable, as all reviewers are given the same version of the inspected artefact.

The models handle variations in the ability of the reviewers to find faults, as well as the faults' probability to be found. The most basic model (M0) assumes that all faults are equally probable to be found and that all reviewers have equal abilities to find faults. More advanced models use either the assumption that the probabilities of fault detection vary (Mh), or the abilities of reviewers vary (Mt), or both (Mth) (see Table 1). Within each model, a number of estimators has been developed.

[1] http://www.iee.org/Publish/INSPEC/.
[2] http://ieeexplore.ieee.org.
[3] http://www.sciencedirect.com.
[4] http://www.acm.org/.

Table 1
The models, prerequisites and estimators in capture–recapture. More models exist in capture–recapture but have not been used for software inspections

Model	Prerequisites	Estimators
M0	All faults have equal detection probability All reviewers have equal detection ability	M0–ML—maximum likelihood (Otis et al., 1978)
Mt	All faults have equal detection probability Reviewers may have different detection abilities	Mt–ML—maximum likelihood (Otis et al., 1978) Mt–Ch—chao's estimator (Chao, 1989)
Mh	Faults may have different detection probabilities All reviewers have equal detection ability	Mh–JK—Jackknife (Burnham and Overton, 1978) Mh–Ch—chao's estimator (Chao, 1987)
Mth	Faults may have different detection probabilities Reviewers may have different detection abilities	Mth–Ch—chao's estimator (Chao et al., 1992)

Mh stands for model with heterogeneity, which refers to the biological context where the population is heterogene, i.e. it is easier to find some animals and harder to find other ones. Mt stands for model with time response which refers to that there may be different conditions for trapping the animals at different trapping occasions, depending on for example, weather conditions. The trapping occasions mean in the software context different reviewers, and hence Mt refers to variability between reviewer abilities.

In addition to capture–recapture, two other fault content estimation methods have been developed that utilize the overlap information, curve-fitting models and subjective estimations. The curve-fitting models use a mathematical function, which is fitted to the inspection data and extrapolated to a limit value. For example, the detection profile method (DPM) (Wohlin and Runeson, 1998), uses an exponential function. Subjective estimations use the reviewers' knowledge to estimate fault content after inspections. The reviewers estimate the most probable value of the number of faults left (El Emam et al., 2000). Enhanced methods have been developed (Biffl, 2000), where reviewers' estimates are combined. Some of these models require that the most probable, minimum and maximum values are estimated by the reviewers.

3. State-of-the-art

This section summarizes and classifies the papers written in the area of capture–recapture applied to software inspections (see Table 2). Research contributions can from a general point of view be divided into *theory*, *evaluation* and *application*. *Theory* includes basic research, which investigates and describes the fundamentals of capture–recapture models and estimators. This has been extensively investigated in biostatistics, but has been further developed and transferred to software engineering and especially to inspections. The second step in a research chain is to *evaluate* the proposed methods and to improve them. The third step in the chain is *application*, where the results from theory and evaluation are transferred to be used in an industrial setting. Moving from theory to application takes a long time (Redwine and Riddle, 1985). Researchers need to establish basic results before software organizations are willing to adopt them. In this survey, only one paper has been classified as an experience paper. That paper only uses results from the very first paper on capture–recapture in software engineering (Eick et al., 1992) and no later research.

A secondary classification is made to classify the papers in subsets of the main classification. Only the subsets needed for the summarized papers are included, i.e. the classification scheme is not necessarily exhaustive. The classification shows the main ideas of the papers and is intended to guide the readers to help them understand the research conducted. The research findings are further discussed in Section 4, where future research is pointed out. The classification of a paper includes a primary classification denoted with an "×" and a secondary classification, denoted with an "(×)". Although a paper contributes in one area, it may also contain smaller contributions in other areas. These smaller contributions receive a secondary classification.

3.1. Basic theory

Most of the basic capture–recapture theory as well as the derivation of all the models and estimators have been described and developed within the research of biostatistics. Capture–recapture in software inspections is an adaptation of an old technique into a new application area. There are, however, some papers published within the software inspection community, which contribute to the investigation and evolution of the basic theory. This includes theory concerning the assumptions or the introduction of new theoretical concepts that arise because of inspections being a new area of application. The report by Freimut (1997) is included in this category because of being the first comprehensive description of all capture–recapture models suitable to be evaluated for use in software inspections.

Table 2
The classification of the capture–recapture papers

Author(s)/year	Theory		Evaluation			Application	
	Basic theory	New approaches	Evaluation of estimators	Improvements of estimators	Estimators and PBR	Tool support	Experience reports
Eick et al. (1992)	×						
Eick et al. (1993)	×						
Vander Wiel and Votta (1993)	(×)		×	(×)			
Wohlin et al. (1995)				×			
Briand et al. (1997)			×				
Ebrahimi (1997)	(×)	×	(×)				
Freimut (1997)	×		(×)		(×)		
Ardissone et al. (1998)						(×)	×
Briand et al. (1998)				×			
Ekros et al. (1998)	×		(×)				
Runeson and Wohlin (1998)			×				
Wohlin and Runeson (1998)		×	(×)				
Miller (1999)			×				
Petersson and Wohlin (1999a)			×				
Petersson and Wohlin (1999b)				×			
Thelin and Runeson (1999)					×		
Biffl (2000)			×				
Briand et al. (2000)			×				
Petersson and Wohlin (2000)				×			
Thelin and Runeson (2000a)	(×)			×			
Thelin and Runeson (2000b)	(×)			x			
Biffl and Grossman (2001)		(×)	×				
El Emam and Laitenberger (2001)	(×)		×				
Freimut et al. (2001)					×		
Wohlin et al. (2001)			×				
Miller (2002)	×						
Miller et al. (2002)						×	
Padberg (2002)		×		(×)			
Thelin et al. (2002)			×	(×)			

A basic question for capture–recapture in software inspections is whether capture–recapture estimators are appropriate to use for software inspections. The assumptions of capture–recapture is whether (a) reviewers are equal in their ability to find faults, (b) there is a risk that the reviewers cooperate, which violates the assumption of independence between reviewers and (c) the faults are equally difficult to discover. These aspects have been investigated by Eick et al. (1993), Ekros et al. (1998) and Miller (2002). There is no agreement of the result found in this subject. For example, Ekros et al. (1998) state that capture–recapture models are not suitable to software inspections whereas Miller (2002) found that no such empirical evidence can be proved. The result in (Miller, 2002) points in the direction that there is no dependence among reviewers. However, if such dependence should exist, Ebrahimi (1997) has developed an estimation model which do not have this restriction.

A solution to (c) would be to group the faults and estimate each group separately by Mt–ML. This solution was suggested by White et al. (1982, p. 163) and Vander Wiel and Votta (1993). This idea has since then been further elaborated by Wohlin et al. (1995) and Runeson and Wohlin (1998) by using a filtering approach of the faults found.

The accuracy of capture–recapture estimations are affected by the amount of input data. The more faults that are found, and the more reviewers that are used, the more accurate become the estimation results. The models have different degrees of freedom, which also affect the estimation results. This is due to that more degrees of freedom require larger amount of data for the estimates.

The accuracy of capture–recapture estimations are often measured as the relative error, see for example (Briand et al., 2000). However, since capture–recapture estimators are meant to be used for decision-making, the measure relative decision accuracy (RDA) may be more appropriate to use (El Emam et al., 2000; El Emam and Laitenberger, 2001). RDA evaluates how the estimators actually are utilized within the inspection process, in contrast to evaluating on only how close the estimations are to the true value. However, there are some limitations of RDA, and one of these is that a threshold needs to be chosen beforehand. The restrictions are further discussed by Thelin and Runeson (2000a).

3.2. New approaches

During the years of research on capture–recapture applied to software inspections, new ideas and suggestions have emerged that reach beyond the mere application and evaluation of existing estimators. This aim is important since the conditions of capture–recapture in software inspections are different from the conditions in biological settings. Most of the capture–recapture estimators have been developed in biostatistics. Hence, these models are fitted for biology problems and not for software inspections. A number of approaches have been developed specifically for software inspections due to that observations of the assumptions are unlikely to be valid. The one presented by Padberg differs somewhat to the others in that it relies more on experience data than on the overlap information in the inspection data (Padberg, 2002).

Ebrahimi (1997) argues that in the software development environment, some degree of collusion among reviewers cannot be avoided. He presents an estimator that does not have the restriction of independence among reviewers. Ebrahimi compares the estimator to Mt–ML and concludes that they produce similar results. However, the estimator needs to be evaluated and compared to other estimators when applied to other data sets.

Another new approach is curve-fitting methods (Wohlin and Runeson, 1998), which basic idea is similar to the ones of reliability growth models for estimation software reliability (Musa, 1998). Wohlin and Runeson (1998) sort and plot the inspection data according to certain rules and then fit a mathematical function to the data. Two methods are proposed, the DPM and the cumulative method. In DPM, the plot shows the number of reviewers that found a specific fault sorted in decreasing order, while the cumulative method plots the cumulative sum of faults that are found.

Biffl and Grossman (2001) investigate an approach which utilizes the information from a second inspection cycle (reinspection). They investigate how estimations from two consecutive inspection sessions should be combined to gain the most accurate estimate. The best approach was to make one estimate from the combined data of the two inspection sessions.

The above described investigations have been classified as new approaches. Only the DPM approach has been replicated by other researchers.

3.3. Evaluation and improvement of estimators

An important part of the capture–recapture research has been to evaluate (a) estimators designed in biostatistics research (b) new proposed estimators and (c) improvements and variants of estimators. This has resulted in a number of papers that evaluate the estimators. As the research has matured, most of the capture–recapture research indicates that Mh–JK seems to be the best estimator for software inspections. In addition, some proposed improvements are evaluated in the papers in this section, e.g. DPM, but still many of these improvements need to be replicated by other researchers. The evaluations and improvements surveyed below refer to a list of aspects: evaluation criteria, number of reviewers, experience-based methods, curve fitting models, decision making, subjective estimates, confidence intervals, filtering, and model selection procedures.

In the evaluation of estimation capabilities, a number of evaluation measures are used:

- Mean relative error—the average of the difference between estimate and true value, sometimes referred to as "bias".
- Variance of relative error—the variance of the difference between estimate and true value, sometimes referred to as "variance".
- RDA—the rate of correct decisions based on the estimate, related to if they were based on the true value.
- Failure rate—the rate of cases for which no estimate is produced at all.
- Root mean square error—the root of the standard deviation^2 + mean^2 of an estimator.

As with all estimation models, the result depends on the quality of data. The accuracy of the estimations is improved by an increased amount of data to base the estimations on. In this case, i.e. applying capture recapture to software inspection data, the same rules applies. The amount of data from an inspection depends on (a) the number of unique faults that are found, and (b) the number of reviewers that participate. From the accuracy point of view, the best would be to have a large number of reviewers. Some studies have shown that at least four to five reviewers should participate in order to make the accuracy acceptable (Briand et al., 2000; Miller, 1999). Hence, some investigations have been focused to improve the performance of estimations when having few reviewers. To get any overlap there must be data from at least two reviewers.

The estimators of capture–recapture have been evaluated in a number of studies. From the beginning, only the most basic models were compared, and then more and more models were used in the investigations. Although the evaluations have been differently evaluated, a common result can be identified. For four reviewers and more, Mh–JK is the most preferable model (Biffl, 2000; Briand et al., 1997; Briand et al., 2000; Miller, 1999; Thelin et al., 2002), although an initial study has pointed out Mt–ML (Vander Wiel and Votta, 1993). They also point in the direction that most models underestimate. However, this may not be a big problem since false positives are often included in the inspection data, which would increase the estimation result. In fact, two papers point out one subestimator of Mh–JK to be the best (Miller, 1999; Thelin et al., 2002).

For few numbers of reviewers, the result is ambiguous. Two investigations have focused specifically on *two* reviewers. These are reported as one simulation study (El Emam and Laitenberger, 2001) and a study with software inspection data (Wohlin et al., 2001). El Emam and Laitenberger (2001) conclude that Mt–Ch is the best estimator for two reviewers. Mh–JK, which has shown to be accurate when the number of reviewers is 4 or more, is non-robust in the case with two reviewers and produces underestimates. However, Mt–Ch does not estimate well when software inspection data are used for two reviewers (Thelin et al., 2002). Wohlin et al. (2001) present three variants of experience-based methods and compare them to the capture–recapture estimators. The variants of the experience-based methods are all based on reviewers' effectiveness. The experience-based methods have less bias than any of the capture–recapture estimators, but not significantly lower. The absolute bias of the relative error as well as the standard deviation is around 20% for the experience-based models. Consequently, if capture–capture should be used for only two reviewers, more data need to be collected. The data collected can be used as input to the experience-based methods presented by Wohlin et al. (2001).

The importance of data collection has been stressed for software inspections since Fagan's paper (1976). Also Gilb and Graham (1993) point out that "The metrics themselves are the lifeblood of inspections.". The possibility of collecting inspection process metrics should be utilized for fault content estimations as well. The use of experience data is shown by Petersson and Wohlin (1999b, 2000), who utilize experience data for capture–recapture estimators as well as for DPM. The general result from these is that experience data improve the estimation results. In addition, Padberg's approach of estimating the fault content is completely based on improvement from experience (Padberg, 2002).

Along with the estimation models from biostatistics, the curve-fitting models presented in Section 3.2 have been evaluated and improved. A problem with all estimators is that they have a tendency to produce extreme under/over estimations. Briand et al. (1998) evaluate a selection procedure that, based on certain criteria, chooses between using an enhanced version of DPM (EDPM) and Mh–JK. This approach shows a small overall improvement of reducing the outliers. However, in the replication by Petersson and Wohlin (1999a), the results are not confirmed. Thelin and Runeson (2000a) extend the number of curve fitting methods. Both linear, quadratic, potential as well as DPM and an extended variant of exponential curves are evaluated and benchmarked against Mh–JK. They conclude that DPM is the best curve fitting method. However, Mh–JK estimates most accurately. Petersson and Wohlin (2000) investigate two alternative ways of improving DPM. The alternatives are to (a) have a variable point of estimation limit that is calculated from historical data, or (b) use the derivative of the curve to determine where to select the estimate. The derivative DPM managed to improve the original DPM but is no improvement compared to Mh–JK.

A purpose of making estimations of the number of remaining faults is to provide information of whether a reinspection should be conducted or not. If a reinspection is made, increased knowledge of the document is gained. The question is then how to best utilize this information in order to get the best estimate of the number of remaining faults. Biffl and Grossman (2001) suggest and evaluate three different formulae for this purpose. The approaches are either to (a) first combine the data from the inspections and then estimate, (b) add the number of faults detected in the first inspection to an estimate of the reinspection or (c) estimate the first inspection and the reinspection separately and then add their results. The best approach is (a), which improved the estimators significantly. The interpretation of the results is that the more time is used in inspection, the more accurate estimation results are obtained.

The aim of applying capture–recapture estimators is to introduce an objective measure to the decision process regarding what to do next with the inspected document. As it is today this decision is based on a more subjective evaluation of the document made by the inspection manager. This evaluation is valuable and capture–recapture based estimates should not aim to replace such evaluation but to complement it. The estimation calculated by the capture–recapture estimators is based on data taken from what faults each reviewer found. The reviewer as a source of information should not be reduced to only a list of found faults but the feeling the reviewer has of the document after having spent time with it during the inspection should be captured too. There have been promising investigations of how this subjective information can be captured in an estimate, which together with the objective support by the capture–recapture estimate can guide the inspection manager even better in his decision. El Emam et al. (2000) present the idea of subjective estimations for software inspections. Biffl (2000) and Thelin (2002) further investigate the concept and evaluate capture–recapture estimators against subjective estimations. Each reviewer gave three estimates: maximum, minimum and most likely number of faults left. Biffl's result shows that subjective estimations have smaller bias, but larger variance. However, Thelin's result shows that capture–recapture is significantly more accurate considering both point estimates and confidence intervals.

Confidence intervals are used to gain more information from a single estimation. A confidence interval consists of an upper and lower limit, and a figure showing how probable it is that the true value lies within these limits. In the capture–recapture research applied to software inspections, most studies have concentrated on point estimates and not on confidence intervals. The confidence interval gives increased information to base decisions on. The discussion of whether to re-inspect or not can be expanded with arguments about worst case/best case scenarios and simple risk calculations based on the probability value. Confidence intervals used for capture–recapture estimators have been considered in some research papers. Vander Wiel and Votta (1993) evaluate Walds Likelihood confidence interval and a confidence interval for Mh–JK (Burnham and Overton, 1978). Vander Wiel and Votta recommend using the Likelihood confidence interval instead of Walds and Mh–JK's, since the former includes the correct number of faults in most cases. However, the Likelihood confidence interval is too conservative, which often leads to wide intervals. Other approaches are mentioned by Biffl (2000) and Biffl and Grossman (2001), but they conclude that these approaches do not work very well. In a recent study by Thelin et al. (2002), both the use of normal and log-normal distributions for the intervals are investigated. They conclude that log-normal distributions are the best alternative when creating confidence intervals.

For point estimates as well as confidence intervals, the estimation models are based on assumptions that reviewers do not find faults totally by random. It is assumed that there is a connection between (a) the ability of the reviewers and the difficulty of the faults and (b) the probability of that a specific fault will be found. Further, it is also assumed that the software inspection data provide information on all faults in the document, including the ones not found. The problem is that less information is available on the set of faults that are difficult to find than on the faults that are easy to find. Moreover, no information is given on the faults that none of the reviewers found. An idea to group the faults to improve Mt–ML's was mentioned by Vander Wiel and Votta (1993). A solution to this problem would be to filter the faults into different groups and estimate each group separately (White et al., 1982, p. 163). This kind of filtering or grouping has been tried by Wohlin et al. (1995) and Runeson and Wohlin (1998). Wohlin et al. identify two possible ways of creating a filter that divides the found faults into groups. Filter 1 is based on the percentage of the reviewers that found a fault. Filter 2 selects all faults only found by one reviewer and then estimates the other faults separately. Faults found by only one reviewer are multiplied with an experience-based factor, since no overlap exists. Applying these filtering techniques, they manage to improve the Mt–ML estimates. The second approach is further investigated by Runeson and Wohlin (1998). They conclude that the mean is not improved with the filtering approach, the variance is smaller using the filtering approach and Mt–ML overestimates in all cases. The filtering approach is criticised in Briand et al. (1998), which show that if an estimator underestimates, the variance increases.

Instead of dividing the inspection data into classes, as in the filtering approach, the characteristics of the data can be tested in order to select the most appropriate estimator. However, to select a model or an estimator is a complicated problem. The more estimators that are used as candidates, the larger is the probability of having one that performs very well. At the same time, with many candidates it becomes harder to choose. On the other hand, having few candidates increases the probability of having only poor estimates to choose from. The aim of selecting estimators to be included in a selection procedure should be to have estimators which complements each other, i.e. when one candidate estimates poorly, there should be another that estimates well. Model selection has been evaluated by the use of distance measures, chi-square tests and smoothing algorithms (Thelin and Runeson, 2000b) and is further analysed by the use of Akaike model selection criterion by Thelin and Runeson (2000a). None of these approaches works appropriately for the data used in software inspections. Furthermore, Briand et al. (1998) suggests and investigates a model selection technique to choose between an enhanced model of DPM (EPDM) and Mh–JK. It showed promising results in the initial study, but was rejected in the replication by Petersson and Wohlin (1999a).

3.4. Estimators and PBR

PBR and capture–recapture have prerequisites that seem to be contradictory. PBR inspections use perspectives applied by the reviewers and the goal is to minimize the overlap between the fault sets that the reviewers find. Since capture–recapture uses the overlap to estimate the fault content, PBR may affect the estimation result. In all investigations, however, the same conclusion is drawn, which is that capture–recapture can be used in combination with PBR with little or no impact on the estimation results.

The three investigations conducted in this area are a simulation study by Thelin and Runeson (1999) and two studies on PBR experiments by Freimut (1997) and Freimut et al. (2001). The common results from the studies are that capture–recapture estimations can be applied even when using PBR and that Mh–JK is the best estimator. One question that remains to be answered is whether the estimators overestimate or not when PBR data are used.

3.5. Experience reports and tool support

Few papers report on industrial use and integration of capture–recapture into an industrial inspection process. As described in previous sections, controlled experiments are performed in industry but they are not included in the application class. The only experience report found is by Ardissone et al. (1998), who describe an implemented tool used for capture–recapture calculations that has been on trial use in an Italian telecom company. The use of the tool is reported to be adopted with encouraging results, but includes little quantitative data.

Miller et al. (2002) describe many capture–recapture estimators in an inspection tool and how it can be used for decision making for project managers.

3.6. Data sets

The above presented studies are based on different data sets. Most of them come from some form of experiments, which is a necessity as the total number of faults in the artefact must be known, to enable evaluation of the estimation capability of the estimators. An overview of the data sets and some characteristics are presented in Table 3.

There are in total 52 data sets used in the surveyed papers. Three of these are simulated data. Twenty six sets come from professional software engineers performing inspections, out of which 16 are from a NASA environment. Twenty three come from an academic environment, with students or faculty performing the inspections. The number of reviewers per data set is in most cases between 5 and 8, while some of the data sets based on students are based on up to 86 reviewers.

The artefacts inspected are requirements documents, design documents, code and pure text documents. The artefacts consist of between 9 and 30 pages for specification documents and 100–300 lines of code for code documents. Artefacts used in the inspections behind many data sets are the automated teller machine (ATM) and the parking garage system documents, in Table 3 referred to as "artificial requirement specification". These artefacts are available in a lab package (Basili et al., 1998).

Different reading techniques are used in the inspections; checklist-based reading, perspective based reading (Basili et al., 1996) and usage-based reading (Thelin et al., 2001a) are defined reading techniques, while ad hoc reading refers to reading based on the reviewers' competence and experience.

In some studies, for example (Briand et al., 1998; Petersson and Wohlin, 1999b), data sets with many reviewers are split into smaller data sets, to provide more data for the evaluation. This is sometimes referred to "virtual inspections". For example, from a set with 10 reviewers, 252 different data sets of 5 reviewers can be constructed. [5]

The dependencies on the data set are illustrated by applying Mh–JK to the different data sets, grouping the reviewers in teams of four. Fig. 1 shows the dispersion of the estimation results, when applying the same estimator to the different data sets. The data sets are NasaAdhAP (1–4), ChklATM (5), EngDMod2 (6), NasaAdhS (7–10), LundPbr (11–12), NasaPbrAP (13–14 and 19–20), NasaPbrS (15–18), IndPbrAP (21–26), Cdata (27–30) (see Fig. 1 and Table 3).

Some of the data sets in Table 3 are grouped into groups since they are derived from the same experiments and used in the same studies. The mapping between studies and groups of data sets is presented in Table 4.

4. Main research findings

The summarized capture–recapture papers have been classified into three main research directions: theory, evaluation and application. The main purpose of this classification is to show how the research has progressed over time. As pointed out by Redwine and Riddle (1985), moving from theory to application takes a long time. Although some issues need to be further evaluated and some more basic research needs to be carried out, the future challenge for capture–recapture researchers is to apply the knowledge gained during the ten years of capture–recapture research for software inspections.

In this section, we extract the knowledge gained from the summaries in Section 3, in order to aid researchers and software organizations with common knowledge collected over the years.

In the *theory* category, the main contribution has been to make the transfer of capture–recapture techniques into the software engineering area and summarize the literature in biostatistics. So far, closed models have been extensively investigated and described.

In the *evaluation* category, many studies are conducted to evaluate the estimators. Evaluations have been made for reading techniques and for different software documents. The common knowledge in this area is:

1. most estimators underestimate,
2. Mh–JK is the best estimator for software inspections,
3. Mh–JK is appropriate to use for four reviewers and more,
4. DPM is the best curve fitting method, and
5. capture–recapture estimators can be used together with PBR.

[5] $\binom{10}{5} = 252$.

Table 3
Characteristics of data sets

Group	Data set	Env.	Doc. type	Insp. tech.	# Rev.	# Faults	Size
	AT&T1	Prof.	Design	?	8	?	?
	AT&T2	Prof.	Req.	?	6	?	?
	AT&Tsim	Sim.	n/a	n/a	5	100	n/a
	Gilb&Graham	Prof.	Design	Chkl	5	27	15 pp
	EngDMod2	Prof./Stud.	Textual	AdH.	22	38	9
NasaAdhAP	AdhAtmJun	NASA	Art. Req.[a]	AdH.	8	29	17 pp
	AdhAtmNov	NASA	Art. Req.	AdH.	6	29	17 pp
	AdhPgJun	NASA	Art. Req.	AdH.	6	27	16 pp
	AdhPgNov	NASA	Art. Req.	AdH.	6	27	16 pp
NasaAdhS	NasaAJun	NASA	Req.	AdH.	7	15	27
	NasaANov	NASA	Req.	AdH.	6	18	27
	NasaBJun	NASA	Req.	AdH.	6	15	27
	NasaBNov	NASA	Req.	AdH.	6	15	27
Myers	Myers	Prof.	Code	AdH	59	15	63 LOC
Cdata	Cdata3A	Prof./Stud.	Code	Chkl	5	22	190 LOC
	Cdata4A	Prof./Stud.	Code	Chkl	5	16	113 LOC
	Cdata5A	Prof./Stud.	Code	Chkl	5	16	85 LOC
	Cdata6A	Prof/Stud.	Code	Chkl	5	35	304 LOC
	Cdata7A[b]	Prof/Stud.	Code	Chkl	4	20	208 LOC
Strath	StrathChklA	Stud.	Code	Chkl	46	12	147 LOC
	StrathChklG	Stud.	Code	Chkl	45	12	141 LOC
	StrathToolA	Stud.	Code	Tool	45	12	147 LOC
	StrathToolG	Stud.	Code	Tool	46	12	141 LOC
	StrathPbrC	Stud.	Art. Req.	PBR	26	26	31 pp
	StrathPbrW	Stud.	Art. Req.	PBR	26	42	24 pp
	StrathChklC	Stud.	Art. Req.	Chkl	24	26	31 pp
	StrathChklW	Stud.	Art. Req.	Chkl	24	42	24 pp
LundPbr	PBRAtm	Stud.	Art. Req.	PBR	15	29	17 pp
	PBRPg	Stud.	Art. Req.	PBR	15	30	16 pp
	ChklATM	Stud./Faculty	Art. Req.	Chkl	6	29	17 pp
NasaPbrAP	PbrAtmJun	NASA	Art. Req.	PBR	6	29	17 pp
	PbrAtmNov	NASA	Art. Req.	PBR	6	29	17 pp
	PbrPgJun	NASA	Art. Req.	PBR	8	27	16 pp
	PbrPgNov	NASA	Art. Req.	PBR	6	27	16 pp
NasaPbrS	PbrNAJun	NASA	Req.	PBR	6	15	27
	PbrNANov	NASA	Req.	PBR	6	18	27
	PbrNBJun	NASA	Req.	PBR	7	15	27
	PbrNBNov	NASA	Req.	PBR	6	15	27
IndPbrAP	PbrStatA	Prof.	Code	PBR	8	19	200–300 LOC each
	PbrStatB	Prof.	Code	PBR	7	19	200–300 LOC each
	PbrTextA	Prof.	Code	PBR	8	16	200–300 LOC each
	PbrTextB	Prof.	Code	PBR	7	13	200–300 LOC each
	PbrZinsA	Prof.	Code	PBR	8	17	200–300 LOC each
	PbrZinsB	Prof.	Code	PBR	7	16	200–300 LOC each
Vienna	ViennaPBR	Stud.	Req.	PBR	86	86	35 pp
	Vienna Chkl	Stud.	Req.	Chkl	83	86	35 pp
	2InspSim	Sim.	n/a	n/a	2	30	n/a
	PbrSim	Sim.	n/a	n/a	4	30	n/a
Ubr	UbrA Basic	Stud.	Design	UBR	14	37	9 pp
	UbrB Basic	Stud.	Design	UBR random	13	37	9 pp
	UbrC Base	Stud.	Design	UBR	11	38	9 pp
	UbrD Base	Stud.	Design	Chkl	12	38	9 pp

[a] Artificial requirement specification.
[b] Only used by Runeson and Wohlin (1998).

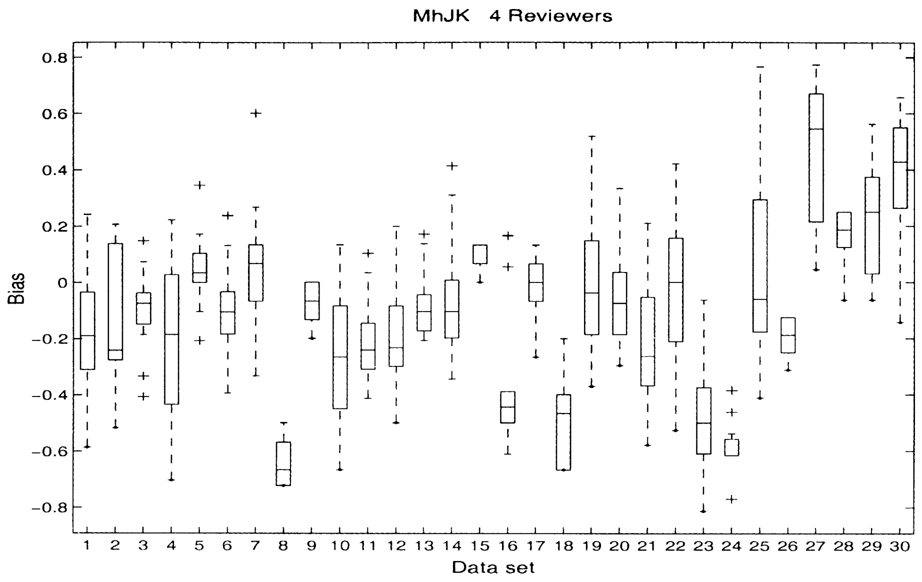

Fig. 1. Example of the characteristics of some of the data sets used in capture–recapture investigations.

In the *application* category, only two papers has been written, although we suspect that more application investigations have been conducted. Still, they need to be reported in order to build a body of knowledge of capture–recapture for software inspections. Another issue worth noting is that the paper classified in the area experience reports does not use any of the research results of the theory or evaluation area other than the initial capture–recapture paper for software inspections by Eick et al. (1992). To address the lack of papers actually reporting the application of capture–recapture, the next section outlines how capture–recapture can be integrated in an inspection process.

5. Application of capture–recapture

Capture–recapture and other fault content estimation methods have been researched in a software engineering context for 10 years. Several lessons have been learned and the methods are sufficiently understood and matured to be introduced into an industrial software development process. The first observation to make is that the actual change in the software inspection process is minor and the potential benefits are an increased ability to estimate and thereby control software quality in terms of software faults.

An example of a process that can be used to combine inspections and capture–recapture is described in this section and shown in Fig. 2.

After the individual inspection (1), the inspection records are handed in to an inspection coordinator who compiles the faults into one document (2). The main purpose of the meeting (3) is to find new faults, but also to share knowledge and experience and take informed decisions about the inspected artefact. The inspection coordinator may then use the inspection record for process improvement and gives the record to the authors of the artefact to correct the faults (4).

The only requirement for the use of capture–recapture methods in an inspection process is that the faults detected should be tracked for each individual reviewer. Studies have shown that the estimation methods are not very sensitive to the actual reading technique (Thelin and Runeson, 1999), which means that the capture–recapture methods can fairly safely be applied independent of if checklists, perspective-based reading or any other individual preparation method is used.

On the other hand, the estimation methods open several opportunities to support the decision-making in the software inspection process. In particular, the capture–recapture methods support two decision points in the inspection process, i.e. after individual preparation, see A in Fig. 2, and after the inspection meeting (if one is held), see B in Fig. 2.

Before decision point A, the process still leaves options with respect to the individual preparation, including:

- Individuals fill in forms that are delivered to one person prior to the meeting. This person compiles the information and is also responsible for making a capture–recapture estimate. Before the estimation, it is important that the person tries to identify which faults noted by the individual reviewers may actually be regarded as the same.
- Each individual fills in a form on-line and the comments from the other reviewers are made available to the reviewers after filling in their form, or after a

Table 4
Data used in different studies

Author(s)/year	At&t1	At&t2	At&t-Sim	G&G	EngD	Nasa-Ad-hAP	Nasa-AdhS	Myers	Cdata	Strath	Lund-Pbr	Chk-lAtm	Nasa-Pbr-AP	Nasa-PbrS	Ind-Pbr-AP	Pbr-Sim	Vi-enna	2Insp-Sim	Ubr
Eick et al. (1992)	×																		
Eick et al. (1993)		×																	
Vander Wiel and Votta (1993)			×																
Wohlin et al. (1995)					×														
Briand et al. (1997)						×	×												
Ebrahimi (1997)		×		×															
Freimut (1997)						×	×						x	x	x				
Ardissone et al. (1998)																			
Briand et al. (1998)						×	×												
Ekros et al. (1998)					×	×	×	×					×	×	×				
Runeson and Wohlin (1998)									×										
Wohlin and Runeson (1998)									×										
Miller (1999)										×									
Petersson and Wohlin (1999a)					×				×		×								
Petersson and Wohlin (1999b)					×	×	×		×		×	×	×	×	×				
Thelin and Runeson (1999)																×			
Biffl (2000)																	×		
Briand et al. (2000)						×	×												
Petersson and Wohlin (2000)					×	×	×		×		×	×	×	×	×				
Thelin and Runeson (2000a)					×	×	×		×		×	×	×	×	×				
Thelin and Runeson (2000b)					×				×										
Biffl and Grossman (2001)																	×		
El Emam and Laitenberger (2001)																		×	
Freimut et al. (2001)																	×		
Wohlin et al. (2001)									×										
Miller (2002)										×									
Miller et al. (2002)										×									
Padberg (2002)						×	×						×	×	×				
Thelin et al. (2002)																			×

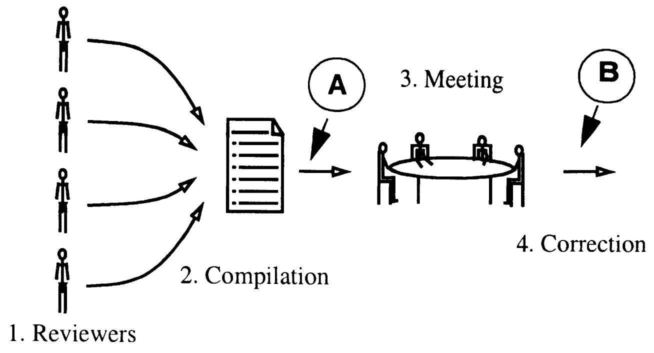

Fig. 2. The inspection process with decision points highlighted.

specific date. This type of system could also easily support capture–recapture estimations and also help reduce the number of false positives, since all reviewers may view the faults marked by the other reviewers.

In the estimation, it is recommended to use the Mh–JK, given the research (e.g. Briand et al., 2000; Miller, 1999; Thelin et al., 2002). It should also be noted that since the estimator has a tendency to underestimate, it is recommended to keep track of the actual behaviour so that any systematic estimation error can be compensated in the long run. This means keeping track of historical data in terms of estimation accuracy. A possible added value is also to introduce subjective estimations (El Emam et al., 2000) regarding the remaining fault content from each individual reviewer. Subjective estimations have shown to be competitive in accuracy, which means that they could provide important input to the decisions in the inspection process.

The subjective estimate, the objective estimate and possibly calibrated objective estimate form an input to a decision point A. The decisions should, of course, not solely be taken based on the estimates. However, they should act as an important source of information for any decision-maker. The decisions could be one of the following, listed from the most negative to the most positive:

A1. Terminate the inspection and send the artefact back for refinement and improvement. This decision basically says that the artefact was not really ready for inspection. This may also be found out by sampling the artefact and using only 1–2 people (Petersson, 2002; Thelin et al., 2001a). Thelin et al. (2001b) describe an inspection process where, in order to save inspection effort, the documents are sampled. Then, based on a pre-inspection on the samples, the inspection effort is focused on the documents that need it the most. Gilb and Graham (1993) and Burr and Owen (1996) 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 (1993) argue that the same types of faults exist in different forms throughout the same document.

A2. An inspection meeting is scheduled based on that the artefact is regarded as being sufficiently inspected by the individual reviewers. A meeting may be needed due to the estimated number of faults remaining or other potential reasons, for example, that the meeting is a means for information sharing and knowledge transfer. This is in most inspection processes the normal procedure, although here it is more viewed as an informed decision.

A3. Assign additional reviewers to further inspect the artefact. This decision indicates that there is an uncertainty in terms of the quality and that more opinions are needed. This basically means postponing the decision, and wait for more input.

A4. Continue the development without a meeting. This decision is based on that the artefact is viewed as having good enough quality and the estimated number of faults is sufficiently low.

A first part of a meeting could be to present the fault content estimations to all reviewers. This could form the basis for a discussion regarding the perceived remaining fault content and it could also act as guidance with respect to try to identify faults at the meeting.

The main part of the meeting is not affected by the use of capture–recapture methods. However, it is recommended at the end of the meeting that the inspection team return to the estimates and try to come to a common view of the remaining fault content. Moreover, it is possible to make a new fault content estimation based on the faults found by the individual reviewers, although with false-positives removed and an improved perception of which faults noted by the reviewers should be viewed as being identical.

The output from the meeting forms the input to the second decision point, i.e. B. The possible output in terms of fault content estimations include subjective individual estimates, capture–recapture estimate before the meeting, subjective estimate by the team at the start and at the end of the meeting and a capture–recapture estimate after the meeting. The input to the decision point includes, of course, also other considerations and experiences, but the opportunity to perform fault content estimations provides an added value to the decision-maker. The five possible decisions at decision-point B are, from the most negative to the most positive decision:

B1. Terminate the current inspection and send the artefact back for refinement and improvement, see also A1.
B2. Assign additional reviewers for further inspection of the artefact. This means going back to decision point A, see also A3.
B3. Update the artefact and then re-inspect it. This may be done by a selected team, possibly smaller than the original team. This decision is not viewed as the same as B1 and it is also viewed as a better situation than B2, since this decision means that there is less uncertainty. Decision B2 is very much a decision taken based on a perception of uncertainty.
B4. Update the artefact and assign one person to check it. After approval, the development continues. In practice, the development may start before the formal approval given that people are informed about the main changes so that the further development is based on a firm ground.
B5. The artefact may need some updating but the task is left for the responsible author. Basically, it means that the artefact should be improved and thereafter it is good enough for the further development.

The objective of this outline of the inspection process with two decision points inserted is to illustrate that capture–recapture methods may make a valuable contribution to any inspection process. Thus, the bottomline is that capture–recapture methods are easily introduced into software inspection processes and that the methods may form an important support for informed decisions.

6. Further research in capture–recapture

In Section 4, the main research findings are discussed in three main areas. Here, these areas are revisited in the light of needed further research.

With respect to *theory* and the models available in biostatistics, there are still areas that have not been explored, e.g. open models and change-in-ratio models (Pollock, 1991). An interpretation of open models for software inspections is that faults may be introduced or removed during inspection, or that estimates are performed based on a series of inspections of the same artefact. This has been regarded as not useful for software inspections, but none has actually explored this subject. Change-in-ratio makes an estimation using the difference between the number of faults in several fault classes over time. This could, for example, be used as risk management information in a spiral or incremental development process to estimate the fault content after design using data from the requirements and the design phase. These are just two topics needed to be looked into. Other future research points are listed below:

- Gain further knowledge of the models of capture–recapture and evaluate whether these are appropriate for software inspections. Some initial research has been conducted in this areas (see Ekros et al., 1998 and Miller, 2002).
- Design and evaluate measures for decision-making in the software inspection process. One such measure is the relative decision accuracy, which purpose is to measure whether the estimators are useful for decision making in software inspections. The decision accuracy has been proposed by El Emam et al. (2000) and is further evaluated by Thelin and Runeson (2000a).
- The estimator developed by Ebrahimi (1997) should be replicated.
- The connection between curve fitting methods, their prerequisites and Mh–JK needs to be investigated.

With respect to *evaluation*, there are some studies improving the estimation results. However, they need to be replicated by other researcher in order to know whether they work in different settings. Main future research points are to:

- Investigate whether one of the subestimators is better than the full estimator of Mh–JK (Miller, 1999). A replication of the initial study by Miller (1999) is carried out by Thelin et al. (2002). However, more replications are needed, especially together with confidence interval investigations.
- Evaluate whether PBR makes estimators overestimate or not (Thelin and Runeson, 1999). In addition, other reading techniques than PBR should be investigated together with capture–recapture. UBR is one reading technique that has to be evaluated together with capture–recapture (Thelin, 2002).
- Replicate suggested improvements (e.g. Briand et al., 1998; Petersson and Wohlin, 1999b; Thelin and Runeson, 2000b).
- Investigate capture–recapture for two and three reviewers (El Emam and Laitenberger, 2001).
- Investigate relation to detection probability, the number of faults found and the number of reviewers with the estimation result (Briand et al., 1998; Thelin et al., 2002).

In the *application* category, there are only two studies published. Consequently, the main research in this area should focus upon transferring fault content estimations into software organizations and report the results as case studies or surveys. This is probably the main challenge for researchers in this field. How should fault content estimations like capture–recapture and subjective methods be applied in a real environment in software organizations? Collaboration between software organizations and researchers will help to improve the estimations for software inspections.

7. Summary

This paper reports the status of capture–recapture research for software inspections, which has been ongoing for 10 years. In this survey, all available papers within the area and their research contribution have been summarized. From these summaries a number of pointers for future research is stated. The papers are also categorized in order to facilitate other researchers' work as well as highlighting the areas of research that need further work.

Three main categories have been used together with a number of sub-categories within each main category.

- Theory—basic theory, new approaches.
- Evaluation—evaluation of estimators, improvements of estimators, estimators and PBR.
- Application—tool support and experience reports.

The most apparent result from the classification is the lack of papers in the application area. Only one published paper reports experience from a trial application of capture–recapture in an industrial environment (not including the controlled experiments).

Furthermore, it is concluded that (1) most of the basic theory is investigated within biostatistics, (2) most software engineering research is performed on evaluation, a majority ending up in recommendation of the Mh–JK model, and (3) there is a need for application experiences. In order to support the application, an inspection process is presented with decision points based on capture–recapture estimates.

Acknowledgements

This work was partly funded by The Swedish Agency for Innovation Systems (VINNOVA), under a grant for the Center for Applied Software Research at Lund University (LUCAS).

References

Ardissone, M.P., Spolverini, M., Valentini, M., 1998. Statistical decision support method for in-process inspections. In: Proceedings of the 4th International Conference on Achieving Quality in Software, pp. 135–143.

Basili, V.R., Green, S., Laitenberger, O., Lanubile, E., Shull, R., Sørumgård, S., Zelkowitz, M.V., 1996. The empirical investigation of perspective-based reading. Empirical Software Engineering 1 (2), 133–164.

Basili, V.R., Green, S., Laitenberger, O., Lanubile, R., Shull, R., Sørumgård, S., Zelkowitz, M., 1998. Lab package for the empirical investigation of perspective-based reading. Available at: <http://www.es.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/manual.html>.

Biffl, S., 2000. Using inspection data for defect estimation. IEEE Software 17 (6), 36–43.

Biffl, S., Grossman, W., 2001. Evaluating the accuracy of defect estimation models based on inspection data from two inspection cycles. In: Proceedings of the 23th International Conference on Software Engineering, pp. 145–154.

Bisant, D.B., Lyle, J.R., 1989. A two-person inspection method to improve programming productivity. IEEE Transactions on Software Engineering 15 (10), 1294–1304.

Briand, L., El Emam, K., Freimut, B., Laitenberger, O., 1997. Quantitative evaluation of capture–recapture models to control software inspections. In: Proceedings of the 8th International Symposium on Software Reliability Engineering, pp. 234–244.

Briand, L., El Emam, K., Freimut, B., Laitenberger, O., 1998. A comparison and integration of capture–recapture models and the detection profile method. In: Proceedings of the 9th International Symposium on Software Reliability Engineering, pp. 32–41.

Briand, L., El Emam, K., Freimut, B., 2000. A comprehensive evaluation of capture–recapture models for estimating software defect content. IEEE Transactions on Software Engineering 26 (6), 518–540.

Burnham, K.P., Overton, W.S., 1978. Estimation of the size of a closed population when capture–recapture probabilities vary among animals. Biometrika 65, 625–633.

Burr, A., Owen, M., 1996. Statistical Methods for Software Quality—Using Metrics for Process Improvement. International Thomson Computer Press.

Chao, A., 1987. Estimating the population size for capture–recapture data with unequal catchability. Biometrics 43, 783–791.

Chao, A., 1989. Estimating population size for sparse data in capture–recapture experiments. Biometrics 45, 427–438.

Chao, A., 1998. Capture–recapture models. In: Armitage, P., Colton, T. (Eds.), Encyclopaedia of Biostatistics. Wiley, New York.

Chao, A., Lee, S.M., Jeng, S.L., 1992. Estimating population size for capture–recapture data when capture probabilities vary by time and individual animal. Biometrics 48, 201–216.

Ebenau, R.G., Strauss, S.H., 1994. Software Inspection Process. McGraw-Hill, New York.

Ebrahimi, N., 1997. On the statistical analysis of the number of errors remaining in a software design document after inspection. IEEE Transactions on Software Engineering 23 (8), 529–532.

Eick, S.G., Loader, C.R., Long, M.D., Votta, L.G., Vander Wiel, S.A., 1992. Estimating software fault content before coding. In: Proceedings of the 14th International Conference on Software Engineering, pp. 59–65.

Eick, S.G., Loader, C.R., Vander Wiel, S.A., Votta, L.G., 1993. How many errors remain in a software design document after inspection? In: Proceedings of the 25th Symposium on the Interface, pp. 195–202.

Ekros, J.-P., Subotic, A., Bergman, B., 1998. Capture–recapture-models, methods, and the reality. In: Proceedings of the 23rd Annual NASA Software Engineering Workshop.

El Emam, K., Laitenberger, O., 2001. Evaluating capture–recapture models with two inspectors. IEEE Transactions on Software Engineering 27 (9), 851–864.

El Emam, K., Laitenberger, O., Harbich, T., 2000. The application of subjective estimates of effectiveness to controlling software inspections. Journal of Systems and Software 54 (2), 119–136.

Fagan, M.E., 1976. Design and code inspections to reduce errors in program development. IBM Systems Journal 15 (3), 182–211.

Fagan, M.E., 1986. Advances in software inspections. IEEE Transactions on Software Engineering 12 (7), 744–751.

Freimut, B., 1997. Capture–Recapture Models to Estimate Software Fault Content, Diploma Thesis, University of Kaiserslautern, Germany.

Freimut, B., Laitenberger, O., Biffl, S., 2001. Investigating the impact of reading techniques on the accuracy of different defect content estimation techniques. In: Proceedings of the 7th International Software Metrics Symposium, pp. 51–62.

Gilb, T., Graham, D., 1993. Software Inspections. Addison-Wesley, UK.

Knight, J.C., Myers, A.E., 1993. An improved inspection technique. Communications of ACM 36 (11), 50–69.

Laplace, P.S., 1786. Sur les Naissances, les Mariages et les Morts Histoire de L'Academic Royale des Sciences, Paris.

Martin, J., Tsai, W.T., 1990. N-fold inspections: a requirements analysis technique. Communications of the ACM 36 (11), 51–61.

Miller, J., 1999. Estimating the number of remaining defects after inspection. Software Testing, Verification and Reliability 9 (4), 167–189.

Miller, J., 2002. On the independence of software inspectors. Journal of Systems and Software 60 (1), 5–10.

Miller, J., MacDonald, R., Ferguson, J., 2002. ASSISTing management decisions in the software inspection process. Information Technology and Management 3 (1–2), 67–83.

Mills, H., 1972. On the Statistical Validation of Computer Programs, Technical report FSC-72-6015, IBM Federal Systems Division.

Musa, J.D., 1998. Software Reliability Engineering: More Reliable Software, Faster Development and Testing. McGraw-Hill, USA.

Otis, D.L., Burnham, K.P., White, G.C., Anderson, D.R., 1978. Statistical inference from capture data on closed animal populations. Wildlife Monographs, 62.

Padberg, F., 2002. Empirical interval estimates for the defect content after an inspection. In: Proceedings of the 24rd International Conference on Software Engineering, pp. 58–68.

Parnas, D.L., Weiss, D.M., 1985. Active design reviews: principles and practices. In: Proceedings of the 8th International Conference on Software Engineering, pp. 418–426.

Petersson, H., 2002. Supporting Software Inspections Through Fault Content Estimation and Effectiveness Analysis, PhD Thesis, Dept. of Communication Systems, Lund University, Sweden.

Petersson, H., Wohlin, C., 1999a. Evaluation of using capture–recapture methods in software review data. In: Proceedings of the 3rd International Conference on Empirical Assessment and Evaluation in Software Engineering.

Petersson, H., Wohlin, C., 1999b. An empirical study of experience-based software defect content estimation methods. In: Proceedings of the 10th International Symposium on Software Reliability Engineering, pp. 126–135.

Petersson, H., Wohlin, C., 2000. Evaluating defect content estimation rules in software inspections. In: Proceedings of the 4th International Conference on Empirical Assessment and Evaluation in Software Engineering.

Pollock, K.H., 1991. Modeling capture, recapture, and removal statistics for estimation of demographic parameters for fish and wildlife populations: past, present, and future. Journal of the American Statistical Association 86 (413), 225–238.

Porter, A., Votta, L., Basili, V.R., 1995. Comparing detection methods for software requirements inspection: a replicated experiment. IEEE Transactions on Software Engineering 21 (6), 563–575.

Redwine, S., Riddle, W., 1985. Software technology maturation. In: Proceedings of the 8th International Conference on Software Engineering, pp. 189–200.

Runeson, P., Wohlin, C., 1998. An experimental evaluation of an experience-based capture–recapture method in software code inspections. Empirical Software Engineering 3 (4), 381–406.

Stringfellow, C., von Mayrhauser, A., Wohlin, C., Petersson, H., 2002. Estimating the number of components with defects post-release that showed no defects in testing. Software Testing, Verification & Reliability 12, 93–112.

Thelin, T., 2002. Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections, PhD Thesis, Dept. of Communication Systems, Lund University, Sweden.

Thelin, T., Runeson, P., 1999. Capture–recapture estimations for perspective-based reading—a simulated experiment. In: Proceedings of the International Conference on Product Focused Software Process Improvement, pp. 182–200.

Thelin, T., Runeson, P., 2000a. Fault content estimations using extended curve fitting models and model selection. In: Proceedings of the 4th International Conference on Empirical Assessment and Evaluation in Software Engineering.

Thelin, T., Runeson, P., 2000b. Robust estimations of fault content with capture–recapture and detection profile estimators. Journal of Systems and Software 52 (2–3), 139–148.

Thelin, T., Petersson, H., Wohlin, C., 2001. Sample-driven inspections. In: Proceedings Workshop on Inspection in Software Engineering, pp. 81–91.

Thelin, T., Runeson, P., Regnell, B., 2001b. Usage-based reading—an experiment to guide reviewers with use cases. Information and Software Technology 43 (15), 925–938.

Thelin, T., Petersson, P., Runeson, P., 2002. Confidence intervals for capture–recapture estimations in software inspections. Information and Software Technology 44 (12), 683–702.

Travassos, G., Shull, R., Fredericks, M., Basili, V.R., 1999. Detecting defects in object-oriented designs: using reading techniques to increase software quality. In: Proceedings of the International Conference on Object-Oriented Programming Systems, Languages and Applications, pp. 47–56.

Vander Wiel, S.A., Votta, L.G., 1993. Assessing software design using capture–recapture methods. IEEE Transactions on Software Engineering 19 (11), 1045–1054.

Weller, E.E., 1993. Lessons from three years of inspection data. IEEE Software 10 (5), 38–45.

White, G.C., Anderson, D.R., Burnham, K.P., Otis, D.L., 1982. Capture–Recapture and Removal Methods for Sampling Closed Populations, Technical Report, Los Alomos National Laboratory.

Wohlin, C., Runeson, P., 1998. Defect content estimation from review data. In: Proceedings of the 20th International Conference on Software Engineering, pp. 400–409.

Wohlin, C., Runeson, P., Brantestam, J., 1995. An experimental evaluation of capture–recapture in software inspections. Software Testing, Verification & Reliability 5 (4), 213–232.

Wohlin, C., Petersson, H., Host, M., Runeson, P., 2001. Defect content estimation for two reviewers. In: Proceedings of the 12th International Symposium on Software Reliability Engineering, pp. 118–127.

Yang, M.C.K., Chao, A., 1995. reliability-estimations & stopping-rules for software testing, based on repeated appearance of bugs. IEEE Transactions on Reliability 44 (2), 315–321.

Håkan Petersson is an Associate Professor at Department of Communication Systems, Lund University, Sweden. He received his PhD in 2002 in Software Engineering and has a M.Sc. in Computer Science and Engineering. His research focus is on software Quality with a special interest in Software Inspections and the estimation of remaining faults.

Thomas Thelin is an Associate Professor in software engineering at the Department of communication Systems, Lund University, Sweden. He has a Ph.D. degree from Lund University 2002, and a M.Sc. in Computer Science and Engineering from the same university in 1997. Thomas Thelin's research interests concern software quality and software process improvement, with special focus on efficient and effective methods for testing and inspection. The research has a strong empirical focus, i.e. is conducted in cooperation with industry. Currently, he works in the European commission project MaTeLo, where the goal is to develop a tool for statistical usage testing using Markov modelling techniques.

Per Runeson is an associate professor in software engineering at the Department of communication systems, Lund University, Sweden, and is the leader of the software Engineering Research Group since 2001. He received a PhD from Lund University in 1998, and a M.Sc. in Computer Science and Engineering from the same university in 1991. He has five years of industrial experience as a consulting expert in software engineering. Dr. Runeson's research interests concern methods and processes for software development, in particular methods for verification and validation, with special focus on efficient and effective methods to facilitate software quality. The research has a strong empirical focus. He has published more than 40 papers in international journals and conferences and is the coauthor of a book on experimentation in software engineering.

Claes Wohlin is professor of software engineering at the Department of Software Engineering and computer Science at Blekinge Institute of Technology in Sweden. Prior to this, he has held professor chairs in software engineering at Lund University and Linköping University. He has a PhD in communication Systems from Lund university. His research interests include empirical methods in software engineering, software metrics, software quality and systematic improvement in software engineering. Claes Wohlin is the principal author of the book "Experimentation in software Engineering—An introduction" published by Kluwer Academic Publishers in 1999. He is co-editor-in-chief of the journal of information and Software Technology published by Elsevier Science. Dr. Wohlin is on the editorial boards of Empirical Software Engineering: An International Journal and Software Quality Journal.