Fraunhofer Einrichtung
Experimentelles
Software Engineering

Quantitative Evaluation of Capture Recapture Models to Control Software Inspections

Authors
Lionel C. Briand
Khaled El Emam
Bernd Freimut
Oliver Laitenberger

IESE-Report No. 053.97/E
Version 1
Dec. 31, 1997

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Quantitative Evaluation of Capture-Recapture Models to Control Software Inspections

Lionel C. Briand, Khaled El Emam, Bernd Freimut, Oliver Laitenberger
Fraunhofer Institute for Experimental Software Engineering
Sauerwiesen 6, D-67661 Kaiserslautern-Siegelbach, Germany
{briand, elemam, freimut, laiten}@iese.fhg.de

International Software Engineering Research Network Technical Report ISERN-97-22

Quantitative Evaluation of Capture-Recapture Models to Control Software Inspections

Lionel C. Briand, Khaled El Emam, Bernd Freimut, Oliver Laitenberger
Fraunhofer Institute for Experimental Software Engineering
Sauerwiesen 6, D-67661 Kaiserslautern-Siegelbach, Germany
{briand, elemam, freimut, laiten}@iese.fhg.de

Abstract

An important requirement to control the inspection of software artifacts is to be able to decide, based on objective information, whether inspection can stop or whether it should continue to achieve a suitable level of artifact quality. Several studies in software engineering have considered the use of capture-recapture models to predict the number of remaining defects in an inspected document as a decision criterion about reinspection. However, no study on software engineering artifacts compares the actual number of remaining defects to the one predicted by a capture-recapture model. Simulations have been performed but no definite conclusions can be drawn regarding the degree of accuracy of such models under realistic inspection conditions, and the factors affecting this accuracy. Furthermore, none of these studies performed an exhaustive comparison of existing models. In this study, we focus on traditional inspections and estimate, based on actual inspections' data, the degree of accuracy of all relevant, state-of-the-art, capture-recapture models for which statistical estimators exist. We compare the various models' accuracies and look at the impact of the number of inspectors on these accuracies. Results show that models' accuracies are strongly affected by the number of inspectors and, therefore, one must consider this factor before using capture-recapture models. When the number of inspectors is below 4, no model is sufficiently accurate and underestimation may be substantial. In addition, some models perform better than others in a large number of conditions and plausible reasons are discussed. Based on our analyses, we recommend using a model taking into account different probabilities of detecting defects and a Jacknife estimator.

1. Introduction

Inspections are an effective method for reducing software development costs and increasing product quality [1][19][17]. The idea behind inspections is to detect defects before they propagate to subsequent development phases and into the field where they cause high rework expenditures.

In practice, however, it has been shown that the effectiveness of inspections can vary widely [7]. To maximize the effectiveness of inspections, one can reinspect an artifact that is deemed to still have high defect content.

Three approaches can be pursued for making in an objective manner the decision of whether to reinspect. The first requires a comparison with historical norms: a document is reinspected for a second time if the number of defects is significantly different from the historical average [14]. Too many defects would indicate a poor document, and too few defects a poor inspection. With this approach, however, a high-quality document may be reinspected, and a poor-quality document may not be re-inspected if the inspection is performed poorly.

The second approach is to use upper and lower thresholds on the number of defects found per unit of size [33]. The lower limit is set to detect poor quality inspections and the upper limit for detecting low-quality documents. Following this approach, however, raises the risk that inspectors are tempted to only find a passing number of defects regardless of the document's quality.

Furthermore, for both of the above approaches, historical data is required to define a norm or thresholds. In practice, such data may not always be available. These difficulties make the case for exploring the third approach.

The third approach is to use the number of defects in the software artifact to calculate how many defects are remaining. This is then uses as the basis for deciding whether to reinspect. Since it is impossible to know the total number of defects in a system before it has been in operation, it is necessary to build estimation models of the number of defects in a software artifact.

This estimation problem is similar to the problem of estimating animal abundance in biology and wildlife research. For example, knowing the population size of deers is essential for deciding on the number to be released for shooting.

One possible solution to this problem is to use capture-

recapture models: animals are captured, marked and released on several trapping occasions. If an animal bearing a mark is captured on a subsequent trapping occasion, it is said to be recaptured. Based on the number of marked animals that are recaptured one can estimate the total population size using statistical models (estimators). When many marked animals are recaptured, one can argue that the total population size is small and vice versa.

The capture-recapture principle in biology can be transferred to inspections: each inspector draws a sample from the population of defects in the inspected software artifact. A defect discovered by one inspector and rediscovered by another is said to be recaptured. Based on estimators similar to the ones used in biology, the total number of defects in the software artifact can be estimated.

Thus far, no study on software engineering artifacts compares the *actual* number of defects to the one predicted by a capture-recapture model. Therefore, in this paper we evaluate the performance (accuracy, variability and failure rate) of different capture-recapture models and their estimators using real software engineering data. Based on this evaluation we make recommendations on which models and estimators to use and under which circumstances.

Briefly, our results indicate that the capture-recapture models and their estimators that we evaluated tend to underestimate the number of defects, with underestimation being largest when there are less than four inspectors. Furthermore, we recommend using a model taking into account different probabilities of detecting defects and a Jacknife estimator when the number of inspectors is at least four.

This paper is organized as follows. In Section 2 we provide a survey of capture-recapture models, and their application in software engineering. Section 3 describes how we have evaluated different capture-recapture models. In Section 4 the results of the evaluations are presented, as well as recommendations on which capture-recapture models and estimators to use. Section 5 discusses the results, and concludes the paper with directions for future work.

2. Review of Capture-Recapture Models and Their Application

This section provides the basic concepts of capture-recapture models, a discussion of the applicability of existing models for software inspections, and a review of the work that has already been performed in a software engineering context.

2.1 Basic concepts

In biology, capture-recapture studies are used to estimate the size of an animal population. In doing so, animals are captured, marked and released on several trapping occasions. The number of marked animals that are recaptured allows one to estimate the total population size based on the samples' overlap. As an example for such an estimation procedure, consider the following (see for example [13]): suppose one wants to estimate the size N of a population that does not change over time, i.e., no animals enter or leave the population. On a first day n_1 animals are captured. These animals are marked somehow and released into the population. After allowing some time for the marked and unmarked animals to mix, a second trapping occasion is performed on a second day. On this day n_2 animals are captured. This sample of n_2 animals consists of m_2 animals bearing a mark (and thus being captured on both days) and u_2 animals without a mark (thus being newly discovered animals). Assuming that the ratio of marked to total animals in the second sample is equal to the ratio of marked to total animals in the entire population, one can derive the so-called Lincoln-Peterson estimator for the number of animals in the population ([30][32]):

$$N = \frac{n_1 \cdot n_2}{m_2} \qquad \text{(eq. 1)}$$

The idea behind using capture-recapture models for software engineering inspections is to let several inspectors draw samples from the population of defects. Based on the overlap of defects between inspectors, one can estimate the number of defects remaining in a software artifact. By subtracting the number of defects that were actually found during inspection from the estimated number of defects, one can calculate the number of remaining defects. Taking into account the number of remaining defects, one can objectively decide whether the software artifact has to be reinspected.

2.2 Applicability of Models for Software Inspections

Capture-recapture models make certain assumptions that differ between biology and software engineering inspections. Thus, before using the models it is necessary to investigate the various assumptions in biology and assess their validity for inspections.

The basic Lincoln-Peterson estimator makes a number of assumptions as follows:

(a) *Number of trapping occasions: only two trapping occasions are performed.* The number of trapping occasions refer to the number of inspectors in

inspections. For inspections, however, we often want to include more than two inspectors.

(b) *Closure: no animals must leave or enter the population during the study*. The number of animals in the population is equivalent to the number of defects in a software artifact. For inspections the assumption of closure is valid since all inspectors inspect the same software artifact and thus face the same population of defects.

(c) *Capture probability: all animals are equally likely to be caught in each trapping occasion.* The capture probability refers to the defect detection probability in inspection. This assumption requires the same detection probability for all inspectors as well as the same detection probability for all defects. For inspections, both assumptions may be violated due to the variation in inspectors' ability (due to experience, education, or reading technique used) as well as defects that are easier to detect than others.

For practical purposes, the first two assumptions can be relaxed. For instance, more than two trapping occasions can be managed by calculating a weighted mean [4]. Also, the Lincoln-Peterson estimator can still be applied when the assumption of closure does not hold: when animals enter or leave the population, only an estimate for the population on the second trapping occasion is provided. It is the third assumption that requires most attention.

Various models (and estimators) have been developed and proposed to alleviate the effects of these assumptions (see [26] for an overview). The most important models one can consider for inspections have been described by Otis et. al. [25] and White et. al. [32]. They present a set of closed models that can deal with more than two inspectors, and that allow for a varying defect detection probability:

(a) Model M0 - No variation: All different defects have the same detection probability. All inspectors have the same detection capability.

(b) Model Mh - Variation by heterogeneity: Different defects can vary in their detection probability. All inspectors have the same detection capability. For instance in [33] it is reported that inspectors often classify defects as easy or hard to detect. This can be considered in this type of models.

(c) Model Mt - Variation by time response: All different defects have the same detection probability. The inspectors have different detection capabilities. Hence, with this variation a model allows for inspectors with differing "general ability". Note, that this "general ability" affects all defects.

(d) Model Mth - Two sources of variation are combined: time response and heterogeneity. This allows for different detection probabilities for different defects and inspectors.

In addition to these sources of variation, Otis et. al. and White et. al consider variations due to behavioural or trap response. This reflects the fact that an animal may change its behaviour due to the process of being captured and marked. For example, when using baited traps, the probability to get caught for the first time is less than the probability for subsequent captures. This is because animals can get fascinated by traps, so marked animals are more likely to get caught than unmarked animals [26]. In inspections, this may be usable to model the fact that defects captured by more than one inspector have usually a higher probability of being detected. However, the estimators for this source of variation depend on the order of trapping occasions (i.e., inspectors). Since no ordering of inspectors seems reasonable in the context of inspections, this estimator is not considered adequate. We summarize the models that we use in Table 1.

Model	Source(s) of Variation
M0	Defects are equal with respect to their probability of being detected, the probability to detect defects among inspectors is the same.
Mt	Defects are equal with respect to their probability of being detected, the probability to detect defects among inspectors varies.
Mh	Defects have different probabilities of being detected, the probability to detect defects among inspectors is the same.
Mth	Defects have different probabilities of being detected, the probability to detect defects among inspectors varies.

Table 1: Relevant Capture-Recapture Models.

2.3 Studies of Capture-Recapture Models in Software Engineering

2.3.1 Existing Studies

The first use of capture-recapture in software engineering was due to Mills [23]. He proposed an estimation of defects in a system based on defect discoveries in the testing phase. His approach was to seed pseudo defects before testing. During testing a tester detects pseudo defects and real defects. Applying the Lincoln-Peterson estimator (eq. 1) to the number of seeded pseudo

defects, the number of detected pseudo defects, and the number of detected real defects gives an estimate for the number of total defects. However, using the Lincoln-Peterson estimator requires the seeded and real defects to have the same detection probabilities.

Based on Mill's approach, several people have used capture-recapture models with seeded defects for estimating software reliability. However, according to Musa et al. [24], this fails to be accurate due to the difficulties to seed the software with defects similar to those naturally occurring. He argues that seeded defects are much easier to find.

A similar approach was introduced by Basin [3][29]. Like Mills' approach, his estimation was based on the Lincoln-Peterson estimator. But instead of seeding defects and one tester, Basin used two testers. The defects detected by the first tester were regarded as "marked defects" for the Lincoln-Peterson estimator.

The first application of capture-recapture methods for inspections was described by Eick et al. [15]. They use capture-recapture models during inspections to predict defect content in the design phase. Like Basin's approach, no artificial defects are seeded into the inspected document. Instead, prior to each inspection meeting, the inspectors search independently for defects. Eick et. al. used the model Mt for defect prediction. Since they had no software artifact with a known number of actual defects, Eick et. al. asked the inspectors for their intuitive opinion about the plausibility of Mt's estimates. The result was that the estimations were consistent with the inspectors' intuition, i.e., a software artifact with a low estimated number of defects was considered to contain a low number of defects.

Since in software engineering defects vary with respect to their defect detection probability, Vander Wiel and Votta [33] compared the model Mt with a model which allows for different detection probabilities of defects, i.e., the model Mh (see Table 1). They performed a Monte-Carlo simulation to investigate the accuracy of two estimators for these models. They observed that Mt performed better than Mh and can be improved by grouping defects into classes wherein the assumption of equal detection probability for different inspectors is more justifiable.

Based on these findings Wohlin et. al. [34] propose two classification techniques, referred to as "filters", that group defects into classes to improve the accuracy of the models. They propose a "percentage filter" and an "extreme filter" that are defined based on experience. With the percentage filter, given a percentage value x, the defects are divided into two classes. The first class contains defects found by more than x% of the inspectors and the second class all defects found by less than x% of the inspectors. For the extreme filter, all defects found by exactly one inspector are put into one defect class, and the remaining in a second class.

These filters were tested in an experiment. However, instead of using software artifacts, the inspectors read a document with grammatical and spelling errors. Wohlin et. al. report that when estimating without grouping, the number of defects was underestimated. Using their classification scheme, it was concluded that a filter may improve the estimates of model Mt.

The current state of knowledge about capture-recapture models for software inspections can be improved by expanding on the scope of previous studies. Specifically, three issues can be addressed, and these are the focus of our study: type of data, impact of the number of inspectors, and using different models and estimators.

2.3.2 Type of Data

So far, no evaluative study of capture-recapture models for inspections was performed on real software artifacts *and* with a known number of defects. We consider this very important if we are to make informed decisions about using capture-recapture models in practice. Thus, we investigate the accuracy of capture-recapture models and their estimators using real software engineering data.

2.3.3 Number of Inspectors

One important element of applying capture-recapture models is the number of inspectors involved. The more inspectors can be included, the more information can be used for estimation. However, using a large number of inspectors is usually not feasible for practical purposes.

Therefore it is necessary to assess the impact of the number of inspectors on the performance of capture-recapture models. Yet, so far evaluations taking this into account have not been done.

In texts dealing with the biological application of capture-recapture models, a number of 5 trapping occasions (equivalent to 5 inspectors in software engineering) is recommended as a rule of thumb, though a number of 7 or 10 was deemed more appropriate [25][32]. However, no quantitative justification or evidence is provided.

In the inspections literature, the reported number of inspectors that typically take part in inspections varies. Bisant and Lyle [5] have found performance advantages in an experiment with two persons: one inspector and the author. Weller presents some data from a field study using three to four inspectors [31]. Bourgois presents data showing that the optimal size is between three and five people [6]. Such a variation and the current lack of quantitative evidence on the impact of the number of inspectors warrants a thorough investigation.

2.3.4 Evaluating Different Models and Estimators

So far in software engineering inspections only the Maximum Likelihood Estimator (MLE) for Model Mt and the Jackknife estimator for Model Mh have been considered. Other estimators present features that seem interesting for inspections. First of all, it is interesting to look at a model that incorporates both different probabilities to detect defects and different detection probabilities for inspectors (Mth). Furthermore, other estimators derived by Chao were developed for situations in which many defects were detected only once or twice. Therefore, they may be appropriate when performing inspections with a few inspectors. For the models in Table 1 we investigated the estimators given in Table 2.

Model	Estimator	Notation
M0	MLE [25]	M0
Mt	MLE [25], Chao's Estimator [9]	Mt MtCh
Mh	Jackknife Estimator [8], Chao's Estimator [10]	Mh MhCh
Mth	Chao's Estimator [11]	MthCh

Table 2: Relevant estimators.

3. Research Method

3.1 Data Set

The data that we use for our evaluation comes from experiments to assess different reading techniques for inspections. The experiments were performed between 1994 and 1995 at the NASA/Goddard Space Flight Centre (NASA/GSFC) [2].

3.1.1 Type of Software Artifacts

The artifacts under study here are requirements documents. Two different sets of requirements documents were used:

- Two generic documents developed for educational purposes. These were the requirements for an automated teller machine (ATM) and a parking garage system (PG). The ATM document was 17 pages long and contained 29 defects. The PG document was 16 pages long and contained 27 defects.
- A NASA/GSFC document consisting of two functional specifications for satellite ground support software. The requirements specifications were structured according to the IEEE standard [18] and the different requirements were stated in natural language.

The defects in the generic documents were not seeded but introduced while developing these documents. The defects of the NASA documents were detected during subsequent development phases.

The four different documents were inspected in two experimental runs each. Since the documents were modified for the second run, we treat both runs independently (i.e., we treat them as eight different documents).

3.1.2 Inspectors

The inspectors were software professionals at NASA/ GSFC with various levels of experience in the application domain and the developement techniques used. This can therefore be considered representative of the circumstances in actual projects.

3.1.3 Inspection Process

The goal of the experiment was to compare reading techniques. It focused exclusively on the defect detection step of an inspection process. All the data that is considered here follow an "Ad-hoc" preparation process [16], i.e., no specific reading technique was used. Neither inspection meetings nor corrections to the inspected documents were performed. We support the view of Sauer et. al. [28] that defect detection is rather an individual than a group activity and that the synergy effect of inspection meetings is rather low in terms of defects that are detected in the meetings. Recent experimental studies [22][20] support this view.

3.2 Evaluation Criteria

To evaluate the different models and estimators, we use three criteria as follows:

- *Relative Error*
  To evaluate the accuracy of the estimate, we use the relative error (RE) defined as:

$$RE = \frac{\text{estimated \# of defects} - \text{actual \# of defects}}{\text{actual \# of defects}} \quad \text{(eq. 2)}$$

  RE allows us to distinguish between overestimation (too many defects were estimated, thus, a positive RE is obtained) and underestimation (too few defects were estimated, thus, a negative RE is obtained).

- *Relative Error Variability*
  Besides the RE of the various models, it is also important to look at their RE variability. Variability tells us whether a large variation around the central tendency can be expected, e.g., whether extreme outliers can be produced by the model. We use inter-quartile ranges as measures of variability, and the median as a measure of central tendency.
- *Failure Rate*
  The best estimator cannot be used alone when it fails to yield an estimate in a large number of cases. Hence an important performance measure that should be looked at is the failure rate of the estimators. We define the failure rate as the percentage of estimates that fail.

3.3 Impact of the Number of Inspectors

Since we have a fixed number of inspectors for each document, we vary the number of inspectors by creating "simulated inspections". A "simulated inspection" is conceptually equivalent to choosing a document and a set of inspectors for this document.

To calculate accuracy for inspections with *k* inspectors, we estimated the number of defects for all documents for all possible combinations of *k* inspectors. For example, if for a document a total of 6 inspectors were available and we wanted to investigate inspections with two inspectors, we formed 15 virtual inspections with two inspectors (i.e., $\binom{6}{2}$). We then ran the capture-recapture models for each combination. This is repeated for each of the eight documents.

In some analyses (to be described below) we also need to obtain a single number characterizing the relative error for each document. We call this the *bias* of the estimator for that document. Bias can be expressed for each document as the central tendency across all combinations for that document. This can be, for example, the mean or median. A disadvantage of the mean is that it is sensitive to extreme values or outliers. A first look at the maximum RE values of our data (see Figure 1) shows, that some estimators (especially those derived by Chao) have indeed large maximum values. Therefore, we define bias as the median RE for all combinations of *k* inspectors for that document.

3.4 Selection of the Best Model

For different numbers of inspectors, we wish to make a recommendation on the most appropriate model(s) to use. The decision procedure for selecting the best model(s) is as follows.

To start off with, we determine which of the two estimators for the h-type and t-type models should be used. This can be done using a paired t-test on the *absolute* bias for the eight documents using a 2-tailed test [21]. We use absolute values here because we do not make a distinction between over- and under- estimation. Furthermore, we do not use the absolute RE values for combinations to make the comparisons because combinations are not independent. If there is no difference, then we select the estimator that shows the least number of extreme outliers.

Subsequently, we determine which sources of variation ought to be taken into account: heterogeneity, time response, or both. We would expect as more sources of variation are considered, the absolute estimation bias would decrease. This means that the M0 model is expected to fare worst, and the MthCh model is expected to fare best. For this, we use a one-tailed paired t-test. For all t-tests, we consider an $\alpha = 0.1$ as our significance level in order to retain sufficient statistical power.[1]

4. Results

4.1 Evaluation of Accuracy and Variability

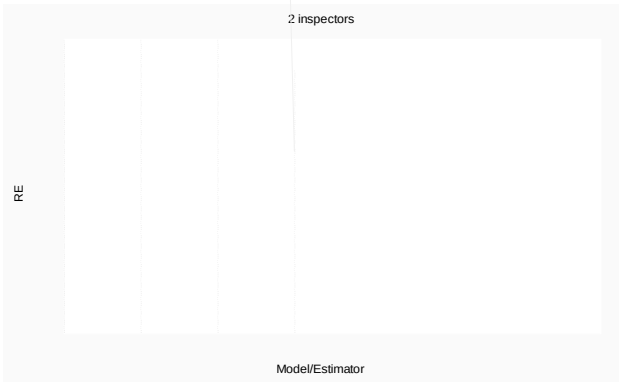
To compare models for a given number of inspectors, both the central tendency of the RE as well as its variability are interesting. The Box-Whisker diagrams for each number of inspectors and all models can be seen in Figure 1.

Results can be summarized as follows:

(a) Generally, there is an obvious trend towards underestimation. The median values consistently underestimate. Underestimation of the number of remaining defects may be substantially more harmful than overestimation since it leads to insufficient effort spent on inspections and poor quality artifacts.

(b) In general, the Chao estimators have relative error which is closest to 0. However, they tend to generate rare but extreme outliers especially for low numbers of inspectors. This limits their practical use if we have no means to control for these extreme overestimations. One possibility is to use several other models/estimators and compare their fault

1. It should be noted that when conducting so many statistical tests, there is a relatively high probability of obtaining one significant result even if all null hypotheses are true. However, we do not use, for example, a Bonferroni adjusted alpha level due to the relatively small sample size used for the t-test.



content estimate with the Chao estimators' estimates. If the latter show to be much larger (say > 50% larger) then they should be considered with care. Also, a comparison with some organizational fault content baseline might help detect unrealistic or extreme estimates.

(c) For less than 4 inspectors, no model yields satisfactory results. The model MhCh shows a relative error closer to 0. However, it exhibits a large variability with extreme outliers. For models with low RE variability, calibration could be considered, i.e., adding a constant percentage defect overhead to each model's estimate.

(d) For 2 inspectors, the estimator for Mh and for Mt show the lowest RE variability. Though they usually underestimate, they might be good candidates for calibration. The model with the smallest median RE value is the Chao estimator for Mh. However, it shows extreme outliers. The model MthCh does not provide an estimate. This is due to the fact that the estimator has a *(k-2)* term in one of its denominators where k is the number of inspectors. Surprisingly, Mt and Mh performed even worse than M0, the simplest model.

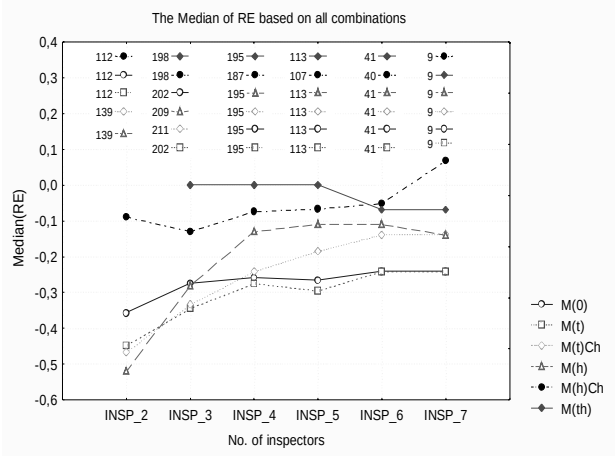


Figure 2: Median RE as a function of the number of inspectors across all 8 documents.

(e) For 3 inspectors, the Jackknife estimator for Mh performed better in terms of RE and RE variability. However, the median RE is still large, i.e. >27%. Therefore, without calibration, this estimator is not likely to be usable in practice.

(f) For 4 and 5 inspectors, the model Mh shows a low median RE. However, it has a relatively large RE variance. Although yielding a low median RE, Chao estimators show very large outliers. The MLE for Model Mt yields a lower RE variability, but tends to underestimate significantly more than Mh. Models MhCh and MthCh seem to be the least likely to lead to underestimation.

(g) For 6 inspectors, the Chao estimators for Mt (i.e., MtCh) and Mth (i.e., MthCh) show the best results. MthCh shows good results in terms of median RE and variability. However, it has a large maximum RE value. MtCh has a smaller maximum but shows poorer values for median RE and RE variabliity. The MLE for Model Mt and M0 underestimates to a great extent. Yet, they show the best behaviour in terms of RE variability. Therefore, they might be possible candidates for calibration.

The median RE as a function of the number of inspectors is shown in Figure 2 for each model. The values at the top of the graph indicate the number of combinations that were generated. It shows that, for most models and over all documents, the median RE decreases fast below 4 inspectors and does not change significantly above that level.

Based on these observations, we can conclude that for inspections with less than 4 inspectors, capture-recapture models are not very accurate. For 4 or more inspectors, the Jackknife estimator for Model Mh and the MLE for Model Mt seem the best suited. However, each of these estimators has its drawbacks. The Jackknife estimator for Model Mh shows smaller median RE but it also shows cases of high overestimation. The MLE for Model Mt underestimates significantly. Yet it shows the lowest RE variability. This model's median RE may therefore be significantly improved through calibration.

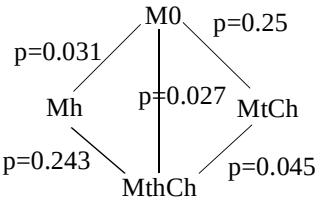


Figure 3: Comparison of models for 4 inspectors.

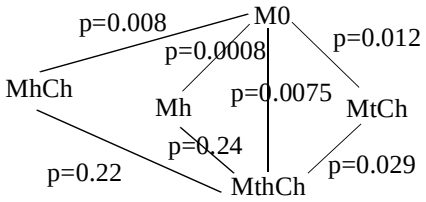


Figure 4: Comparison of models for 5 inspectors.

4.2 Selection of the Best Model

For these results, we focus on 4 and 5 inspectors only since we concluded that for less than 4 inspectors the capture-recapture models become unusable in practice. For six inspectors the number of combinations used to obtain the bias was rather small, and hence is expected to be unstable. This suggests that any conclusions drawn from an analysis using 6 inspectors from our data set would be equivocal.

For 4 inspectors, there was no difference between the two estimators for model Mh (using a t-test). We therefore select the Jacknife estimator since Chao's estimator shows cases of extreme overestimation for 4 inspectors (see Figure 1). Chao's estimator for model Mt, however, was better than the MLE (p=0.05), we therefore select that one.

For 5 inspectors, again there was no difference between Chao's estimator and the Jacknife (p=0.84) for model Mh. Chao's estimator for 5 inspectors does not exhibit extreme overestimation, therefore we keep both estimators for model Mh. Chao's estimator for model Mt was better than the MLE (p=0.02), and therefore we select that one.

The results used for the following discussion are shown in Figure 3 and Figure 4. In these diagrams the nodes are the models and their estimators, and the edges indicate a comparison using the t-test. The p values for each comparison are also given.

The comparisons of different models for 4 inspectors are shown in Figure 3. All means in that figure are in the expected direction (i.e., the models that account for more sources of variation tend to have lower mean absolute bias). As can be seen, the heterogeneity (model Mh with the Jacknife estimator) source of variation improves the mean absolute bias over M0. The time response source of variation (model MtCh) adds no improvement to M0. When two sources) as Mh w7(e)1pahsit merd7(e)17(n)ion i Mh v(T)13(h(e)17(r m)(Mh t-7(s)11( )24(the)]TJ T* 0.05(Mh 25(s22(w)2(

absolue biae af

inspections. Capture-recapture models allow us to estimate the total number of defects in a software artifact. For using capture-recapture models in practice, the performance of various models and estimators must be evaluated.

Below we present a summary of our evaluation findings and recommendations:

- *Number of Inspectors*
  Our results indicated that the number of inspectors does have an impact on the RE, RE variability, and failure rate of capture-recapture models. Specifically, it is suggested that capture-recapture models ought not be used with less than 4 inspectors. An important issue is whether a high number of inspectors can always be usedec″-0.018 12(e)1-28(r)-beT655 Twı″[(i)22(n)0(s)11(p)0(e)-7f0.42(il)21(uv(l)2e)16(r14(118(c

International Software Engineering Research Network, Technical Report ISERN-97-21, 1997.

[8] K. P. Burnham and W.S. Overton: “Estimation of the Size of a Closed Population when Capture Probabilities Vary Among Animals”. In *Biometrika*, 65:625–633, 1978.

[9] A. Chao: “Estimating the Population Size for Capture-Recapture Data with Unequal Catchability”. In *Biometrics*, 43:783–791, December 1987.

[10] A. Chao: “Estimating Animal Abundance with Capture Frequency Data”. In *Journal of Wildlife Management*, 52(2):295–300, 1988.

[11] A. Chao, S.M. Lee, and S.L Jeng: “Estimation Population Size for Capture-Recapture Data when Capture Probabilities Vary by Time and Individual Animal”. In *Biometrics*, 48:201–216, March 1992.

[12] E. Doolan: “Experience with Fagan’s Inspection Method”. In *Software - Practice and Experience*, 22(2):173-182, February 1992.

[13] E. Dudewicz: *Modern Mathematical Statistics*. John Wiley & Sons, Inc., 1988.

[14] S. Eick, C. Loader, M. Long, L. Votta, and S. Vander Wiel: “Estimating Software Fault Content Before Coding”. In *Proceedings of the 14th International Conference on Software Engineering*, pages 59–65, 1992.

[15] S. Eick, C. Loader, S. Vander Wiel, and L. Votta: “How Many Errors Remain in a Software Design After Inspection?” In *Proceesings of the 25th Symposium on the Interface*. Interface Foundation of North America, 1993.

[16] M. Fagan: “Design and Code Inspections to Reduce Errors in Program Development.” In *IBM Systems Journal*, 15(3):182-211, 1976.

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

[18] IEEE Standards Collection, Software Engineering, Std 830-1993, 1994.

[19] C. Jones: “Software Defect Removal Efficiency”. In *IEEE Computer*, vol. 29, No. 4, 1996.

[20] O. Laitenberger and J. DeBaud: “Perspective-based Reading of Code documents at Robert Bosch GmbH”. In *Proceedings of the First Conference on Empirical Assessment and Validation*, March 1997.

[21] L. Lapin: *Statistics for Modern Business Decisions*, Second Ed., Hartcourt Brace Jovanovich, Inc., 1978.

[22] P. McCarthy, A. Porter, H. Siy, L. G. Votta: “An Experiment to Assess Cost-Benefits of Inspection Meetings and their Alternatvies”. In *Proceedings of the International Metrics Symposium*, Berlin, March 1996.

[23] H. Mills: *On the Statistical Validation of Computer Programs*. Technical Report Report FSC-72-6015, IBM Federal Systems Division, 1972.

[24] J. Musa, A. Iannino, and K. Okumoto: *Software Reliability: Measurement, Prediction, Application*. McGraw-Hill, 1987.

[25] D. Otis, K. Burnham, G. White, and D. Anderson: “Statistical Inference from Capture Data on Closed Animal Populations”. In *Wildlife Monographs*, (62):1–135, 1978.

[26] K. Pollock: “Modeling Capture, Recapture, and Removal Statistics for Estimation of Demographic Parameters: Past, Present, and Future”. In *Journal of the American Statistical Association*, 86(413):225-238, March 1991.

[27] J. Rice: *Mathematical Statistics and Data Analysis*. Duxbury Press, 1987.

[28] C. Sauer, R. Jeffery, L. Lau, P. Yetton: “A Behaviourally Motivated Programme for Empirical Research into Software Development Technical Reviews”. Technical Report, Center for Advanced Empirical Research, 1996.

[29] G. Schick and R. Wolverton: “An Analysis of Competing Software Reliability Models”. In *IEEE Transactions on Software Engineering*, SE-4(2):104–120, March 1978.

[30] G. Seber: *The Estimation of Animal Abundance and Related Parameters*. Charles Griffin & Company Ltd., 2nd. edition, 1982.

[31] E. Weller: “Lessons from Three Years of Inspection Data.” In *IEEE Software*, 10(5):38-45, September 1993.

[32] G. White, D. Anderson, K. Burnham, and D. Otis: *Capture-Recapture and Removal Methods for Sampling Closed Populations*. Technical Report, Los Alamos National Laboratory, 1982.

[33] S. Vander Wiel and L. Votta: “Assessing Software Designs using Capture–Recapture Methods”. In *IEEE Transactions on Software Engineering*, 19(11):1045–1054, November 1993.

[34] C. Wohlin, P. Runeson, and J. Brantestam: “An Experimental Evaluation of Capture-Recapture in Software Inspections”. In *Software Testing, Verification and Reliability*, 5:213–232, 1995.

Document Information

Title:	Quantitative Evaluation of Capture Recapture Models to Control Software Inspections
Date:	Dec. 31,1997
Report:	IESE-053.97/E
Status:	Final
Distribution:	Public
	also published as ISERN-97-22