Evaluation of Capture-Recapture Models for Estimating the Abundance of Naturally-Occurring Defects

Gursimran Singh Walia
Mississippi State University
300 Butler Hall, Box 9637
Mississippi State, MS 39762
+1 662-325-8798
gw86@cse.msstate.edu

Jeffrey C. Carver
University of Alabama
101 Houser Hall
Tuscaloosa, AL 35487
+1 205-348-9829
carver@ca.ua.edu

ABSTRACT

Project managers can use capture-recapture models to manage the inspection process by estimating the number of defects present in an artifact and determining whether a reinspection is necessary. Researchers have previously evaluated capture-recapture models on artifacts with a known number of defects. Before applying capture-recapture models in real development, an evaluation of those models on naturally-occurring defects is imperative. The data in this study is drawn from two inspections of real requirements documents (that later guided implementation) created as part of a capstone course (i.e. with naturally occurring defects). The major results show that: a) estimators improve from being negatively biased after one inspection to being positively biased after two inspections, b) the results contradict the earlier result that a model that includes two sources of variation is a significant improvement over models with one source of variation, and c) estimates are useful in determining the need for artifact reinspection.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications, D.2.4 Software/Program Verification, D.2.8 Metrics, D.2.9 Management, and K.6.3 Software Management.

General Terms

Management, Measurement, Documentation, Experimentation, Human Factors, Verification

Keywords

Software Inspections, Capture-Recapture Models, Defect Estimation, Requirements, Validation and verification, Empirical Study

1. INTRODUCTION

Software inspections are an effective approach to improve quality by locating defects early and avoiding their propagation to subsequent phases. However, inspections can only provide information about the defects that are detected and not about those that have yet to be detected. Therefore, to make a decision about the need for re-inspection, developers tend to assume that if a large number of defects were found during the inspection, then a large number probably remains in the document [1]. Because an inspection requires the use of limited resources, Informed, objective estimates of the remaining defects should guide this decision, rather than the subjective estimates or historical trends of the remaining defects.

During software development, project managers and developers need to be able to estimate the number of defects still remaining in the document. Reliable estimates of the remaining defects after an inspection can help managers decide whether to perform a re-inspection or to pass the artifact through to the next phase. Among the different approaches used to make this estimate (e.g., defect density, subjective assessment, historical data, capture-recapture method, curve fitting method), capture-recapture is the most appropriate and widely used method [2, 10].

Capture-Recapture (CR) is a statistical method that was originally developed by biologists to support population size estimates. CR is used in biology by capturing fixed number of animals and marking them as captured. The marked animals are then released back into the population and allowed time to re-mix. Then another trapping occurs. Any marked animals that are captured at the second trapping are said to be recaptured. The total number of animals in the population is estimated based on the number of marked animals that are recaptured. More recaptures indicates a smaller estimated population [15, 22].

Using the same principle, the CR method can be used to estimate the number of defects in a software artifact. During an inspection, each inspector finds, or captures, some defects (analogous to trapping animals). If the same defect is found by another inspector (analogous to another trapping occasion), it is said to be recaptured [2, 8]. The number of defects found and the overlap of defects among inspectors during an inspection is used to estimate the total number of defects present in that artifact. Similar to biology, larger overlap results in a smaller and more precise estimate. The difference between the estimated number of defects and number of defects actually found provides insight into the number of defects remaining. The CR method makes use of

different types of models (with varying assumptions), each with a different set of estimators (described in more detail in Section 2).

Biologists have comprehensively evaluated the use of CR models [15, 22], but their use and evaluation in software engineering is relatively new. Most empirical studies in software engineering have evaluated the use of CR models on software artifacts with a known number of seeded defects [2, 8, 9, 10, 11, 13, 16, 17-18, 20-21, 23-24]. However, in live software development, the actual defect count of an artifact is unknown. So, it is not clear what effect the use of seeded defects had on the estimation results. There is little evidence to support the efficacy of using CR models in real software development (with an unknown number of naturally occurring defects). Moreover, the empirical studies until now have focused on a few selected estimators for each CR model type. A more detailed analysis of all the available estimators for each CR model is needed.

This paper performs a comprehensive evaluation of CR models and their estimators on real software artifacts that contain natural defects made during their development. The artifacts were inspected two times; and we evaluate the performance of the CR models and estimators after each inspection cycle. We compare the performance of the estimators and choose the best model. We also analyze the ability of the CR models to accurately predict the need for a re-inspection. Finally, the findings from this study are compared with findings from previous research in software engineering and biology.

Section 2 describes the CR models and their application to inspections. Section 3 discusses the background literature that inspired this study. Section 4 describes the design of the study. Section 5 describes the data analysis and results. Section 6 discusses the threats to validity. Section 7 discusses the relevance of the results and compares the results with previous results in software engineering and biology. Section 8 concludes the study.

2. USE OF CAPTURE-RECAPTURE IN SOFTWARE INSPECTIONS

CR models in biology and wildlife research make certain assumptions, and only some of them are met in software inspections. The assumptions made by CR models that hold for software inspections are: a) Closed population: All inspectors work independently, inspect the same artifact, and the artifact remains unchanged during the experiment, and b) Defects are recorded on defect lists and maintained to make distinction between captured and recaptured defects. However, the assumption regarding *equal capture probability* is not satisfied [2, 16, 22]. In software inspections, inspectors can have different defect detection abilities (based on their difference in background education, training, or innate ability) and defects can have different detection probabilities (as some defects are easier to locate than others) [2].

To accommodate these variations, four CR models, each with different sources of variation, are shown in Table 1. These models were originally developed by biologist [15, 22], and have been used in previous empirical studies in software inspections [2, 8-9, 13, 16, 17-18, 21, 24]. These studies are discussed in detail in Section 3.

Furthermore, each CR model in Table 1 has different estimators. Each estimator uses a different statistical approach to produce the population estimate. The estimators used in this study are shown in Table 2 along with the associated CR model from Table 1. Some of the estimators have been used in previous studies, while others are used here for the first time in the software inspection domain (those marked with *). The mathematical details of the estimators are beyond the scope of this paper. More details about each CR estimator can be found in the provided references.

All of the CR estimators use the data organized in the same way for computing their estimates i.e., a simple matrix with rows representing defects and columns representing inspectors. An entry in the matrix is 1 if the defect was found by the inspector and 0 otherwise. Each estimator can derive all the other required statistics from this matrix to estimate the defect population [2, 15, 22].

3. EMPIRICAL STUDIES OF CAPTURE-RECAPTURE MODELS IN SOFTWARE INSPECTIONS

Eick et al., proposed the use of CR for software inspections and performed the first study of CR models in software inspections. They used the maximum likelihood estimator (MLE) for the M_t model to estimate the defects remaining in real requirements and design documents at AT&T. The defects were naturally occurring and not seeded. The results showed that the estimates of remaining defects were similar to the subjective opinions of the

Table 2. CR estimators

Models	Estimators
M_o	Unconditional Maximum Likelihood Estimator (M_o-UMLE) [15]
	*Conditional Maximum Likelihood Estimator (M_o-CMLE) [7]
	*Estimating Equations Estimator (M_o-EE) [25]
M_t	Unconditional Maximum Likelihood Estimator (M_t-UMLE) [15]
	*Conditional Maximum Likelihood Estimator (M_t-CMLE) [7]
	*Estimating Equations Estimator (M_t-EE) [25]
	Chaos Estimator (M_t-Ch) [4]
M_h	Chaos Estimator (M_h-Ch) [5]
	Jackknife Estimator (M_h-JK) [3]
	*Sample Coverage Estimator (M_h-SC) [12]
	*Estimating Equations (M_h-EE) [25]
M_{th}	*Sample Coverage Estimator (M_{th}-SC) [12]
	*Estimating Equations Estimator (M_{th}-EE) [25]
	Chaos Estimator (M_{th}-Ch) [5-6]

Table 1. CR models

Model	Source of Variation(s)
M_o	Inspectors are equal in their defect detection ability and defects have equal detection probability
M_t	Inspector varies in their defect detection abilities and defects are equally detectable
M_h	Defects varies in their defect detection probability and inspectors are equally able
M_{th}	Inspectors differ in defect detection ability, and defects differ in detection probability

developers. Based on the inspection results in their environment, Eick et al., recommended that an artifact be re-inspected if the number of remaining defects is greater than 20 percent of the total [8, 9]. This recommendation is still used by CR studies. With the introduction of new estimators, other empirical studies were performed to evaluate and improve them.

Among those early efforts, Weil and Votta used artifacts with seeded defects to compare the performances of the MLE estimator for the M_t model and the Jackknife (JK) estimator for the M_h model when their assumptions were violated. The authors tried to improve the accuracy of these estimators by suggesting a grouping method. The result showed that the MLE estimator was more accurate than the JK estimator, with and without grouping [21]. Wohlin et al., also suggested an improvement to MLE estimator and evaluated it on a text document with seeded defects. Contrary to earlier findings, the results showed that the MLE estimator overestimated the number of defects. However, this study was done with a non-software engineering artifact [23, 24].

Later on, Briand et al., evaluated a series of CR models using data generated from the inspection of artifacts with seeded defects by NASA software professionals. The result from this study showed that the estimators generally underestimate the number of defects, and recommended using the JK estimator. The result also showed that a minimum of four inspectors are needed for achieving satisfactory estimates [2]. This study was later replicated and confirmed that the Mh model is the superior model and recommended the JK estimator [13].

Emam et al. evaluated CR estimators for two inspectors using artifacts with seeded defects and analyzed their ability to accurately determine the need for re-inspection. The results showed that Mh-JK and Mt-Ch are the best estimators, and that not all estimators helped in making correct decisions on re-inspections [11]. Emam, et al., also advocated the use of the inspectors' subjective opinions along with the CR estimates when the models are used during real development (i.e., when the actual defects are unknown rather than seeded) [10]. Another study evaluated the CR methods using artifacts with seeded defects and advocated the use of confidence interval coverage rather than point estimates for achieving more trustable estimates. Conversely, this study showed that the subjective estimates are significantly less accurate than the CR estimates [18].

Previously, we reported on a study concerned with the effect that increasing the number of inspectors has on the quality of the CR estimates, using a software artifact with seeded defects. A major result from this study was the identification of the minimum number of inspectors required for achieving different levels of estimation accuracy [20]. Other researchers (e.g., Runeson, Thelin, and Wohlin, et al.,) have conducted similar CR studies to evaluate the CR estimators with defects (real or artificial) seeded into artifacts before inspection, and can be referred from a list of all the CR studies conducted over ten years (1992-2002) of the research in software inspections [16]. Analysis of these studies acknowledged the fact that the major results regarding the evaluation of CR models are derived from studies conducted on the artifacts with seeded defects.

To summarize, the major results from the evaluation of CR models in software inspections include: a) a consensus that M_h is the best model and M_h-JK is the most accurate estimator, b) estimators generally underestimate, but improve with more

defects and inspectors, c) a minimum of four inspectors are required for achieving satisfactory estimates, and c) there is no consensus on the relative accuracy of subjective and objective estimates.

4. STUDY DESIGN

The earlier CR studies evaluated the four basic models and some of their estimators. A major difficulty when evaluating CR models is that the number of actual defects is not known beforehand. Therefore, many researchers have used seeded defects to allow for comparisons of the estimates to the actual value (i.e., number of seeded defects). Accordingly, the major findings and recommendations are based on inspections using artifacts with seeded defects. No empirical study has comprehensively evaluated the CR models in a case where the number of defects is not known beforehand, as would be the case with any real development.

Furthermore, software reliability research has shown that seeded, artificial defects differ in detection probability from naturally occurring defects and are easier to detect. Even when re-seeding real defects, their densities differ from that of natural occurring defects [14]. Therefore, the nature of the defects can influence the estimation results. To provide better information for project managers and inspectors to use when deciding on the adoption of CR models in their organization, it is important to evaluate the CR models in real settings.

This paper describes a comprehensive evaluation of CR models and their estimators on real software artifacts that were developed by students in a senior-level capstone software engineering class (i.e. they were created to guide the later implementation of the system) with naturally occurring defects, and later inspected in the same environment. In addition, each artifact was inspected twice, which allowed the analysis of the CR estimator's ability to make correct recommendations about the need for re-inspection. The findings from this study are then compared with the earlier findings to gain more insights.

4.1 Goal(s) of the Study:

The main goal of this study is to evaluate the CR models and estimators on real software artifacts with naturally-occurring defects, and number of actual defect count not known beforehand. More formally, the goal is to:

Analyze the CR models and estimators
For the purpose of evaluation
With respect to the ability to estimate the number of remaining defects
From the point of view of project managers and inspectors
In the context of a real requirements document

4.2 Data Set

The data for the CR analysis was drawn from earlier inspection studies conducted at Mississippi State University (MSU). The goal of those studies was to investigate the use of human errors (i.e., mistakes in the thought process) committed during development for improving the quality of the software artifact [19]. Only the information relevant to the CR analysis is provided here.

4.2.1 Software Artifacts and Software Inspectors

Table 3. Artifacts, and Inspectors used in this Study

Artifact	Name	Description	Number of Inspectors	1st Inspection Defects	2nd Inspection Defects	Total Defects
A	Starkville theatre system	Management of ticket sales and seat assignments for the community theatre	8	30	25	55
B	Management of apartment and town properties	Managing apartment and town property, assignment of tenants, rent collection, and locating property by potential renters	8	41	64	105
C	Conference management	Helping the conference chair to manage paper submission, notification of results to authors, and other related responsibilities	6	52	42	94
D	Conference management	(Same as C)	6	64	54	118

Inspection data from four software artifacts is used in this study. The artifacts were developed by senior-level undergraduate students, majoring in either computer science or software engineering enrolled in the Software Engineering Senior Design Course at MSU during the Fall 2005 and Fall 2006 semesters. The course required students to interact with real customers, elicit, and document requirements that they would later implement. So, even though the developers are students, the artifacts are realistic for a small project. The subjects were divided into 4 teams (with 8, 8, 6, and 6 students respectively) that developed the requirement documents for their respective systems as shown in Table 3. (Note that even though artifact D has the same description as artifact C, it was a different set of requirements created by a different set of subjects.) Each artifact was then inspected independently by the same developers who created it [19].

4.2.2 Software Inspection Process

The goal of the original experiments was to investigate the usefulness of error information in software inspections as opposed to just using fault information. The inspection process consisted of having each inspector inspect the artifact using a simple fault checklist and log the faults. After that, training was provided on how to abstract errors from faults, how to classify the errors, and how to use the errors to re-inspect the requirements document for more faults. The same process was used to inspect all four artifacts. Note that the artifacts were not modified or corrected between inspections (i.e. the same artifact was re-inspected).

Therefore, we have inspection data from first inspection, the second inspection, and total for each artifact. Note that the last three columns in Table 3 show total unique defects found during the first inspection, unique defects found during the second inspection (different from the defects found during the first inspection), and total defects for both inspections (the sum of the defects from the first and second inspection) respectively. Because the same process was used for all four artifacts and the subjects were all drawn from the same population, the CR analysis combines the inspection data from all the four artifacts into one large dataset for evaluating the CR models and their estimators. Because each artifact had a different number of defects, the analysis in this paper focuses on percentages of defects found to normalize the data.

4.3 Experiment Procedure

To evaluate the CR models and estimators after each inspection, we used two automated tools (CAPTURE [10] and CARE [3]) to calculate the estimates for each of the fourteen estimators as follows:

a) *Calculate estimates after first inspection:* For each artifact, the defects found during the first inspection (column 5 of Table 3) by all inspectors are inserted into a matrix (as described in Section 2). This matrix is then fed to the automated tools to produce the estimates for all the estimators. Using the estimated defect count and the number of unique defects found at first inspection, the number of remaining defects can be estimated.

b) *Calculate estimates after second inspection*: Because the artifacts were unchanged between inspections, to calculate the estimates after second inspection, we use the total defects found from both inspections (column 7 of Table 3). It did not make sense to use only the data from inspection 2 because some information would have been excluded making the estimates inaccurate. Therefore, for each artifact, the defects found at the first and second inspection by all inspectors are inserted into a matrix. The matrix is then fed to the automated tools to produce the estimates from all the estimators. Using the estimated defect count and the number of defects found after both inspections, the remaining defects after second inspection is estimated.

4.4 Evaluation Criterion

The estimators are evaluated based on their performance after the first and second inspection using these parameters: accuracy (bias), precision (variability), and failure rate.

The *accuracy (bias*) is measured as the relative error (R.E) of an estimate. It is calculated as:

Relative error = (Estimated number of defects – Actual number of defects) / Actual number of defects

A R.E of zero means absolute accuracy. A positive R.E. means an overestimation. A negative R.E means an underestimation. R. E threshold of +/- 20% is considered satisfactory for estimates.

Because we do not know the actual number of defects, the total number of exclusive defects found after both inspections is assumed to be the actual defect count for the purposes of this study. The difference between the estimated defect count and this actual defect count is used to evaluate the accuracy of CR estimators. Furthermore, the error in the estimates is calculated relative to each artifact to allow for combination of the results from all the artifacts.

The *precision (variability)* of an estimator is measured by calculating the inter-quartile range (IQR), the outliers, and the extreme outliers.

The *failure rate* is the number of times an estimator fails to produce an estimate.

5. DATA ANALYSIS AND REPORTING RESULTS

This section first compares the estimates produced after the first inspection and the second inspection. Then, it analyzes the best CR model. Finally, it discusses how to use the CR estimators to manage the inspection process. An alpha value of 0.1 is selected in this initial study, because there are only four data points (four artifacts).

5.1 Comparison of the Estimates after One and Two Inspections

The performance of the CR estimators is compared after the first and second inspection based on the relative error values. Figure 1 and Figure 2 show the accuracy and precision of the each estimator after the first inspection and after the second inspection respectively. Figures 1 and 2 partition the relative error values into different regions: the solid line represents absolute accuracy (0 biases), the lower dashed line is a 20% underestimation, and the upper dashed line is a 20% overestimation. Important observation from Figures 1 and 2 are:

a) In terms of accuracy, there is a general trend that the CR estimators underestimate the defect count after first inspection as most estimators fall below the 0% line and many fall below the -20% region. There is also a general trend that the CR estimators overestimate the defect count
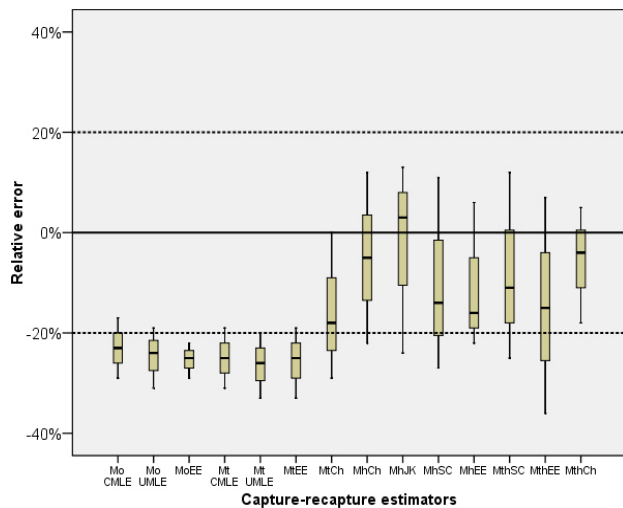
after the second inspection, but, most of the estimates (except M_h-JK) fall within the acceptable range of 0%-20%. In addition, the estimators for M_h and M_{th} models are generally less biased than the estimators for the M_o and M_t models at both inspection cycles (except for the M_h-JK estimate after the second inspection).

b) In terms of precision, the estimators of M_o and the M_t models are generally more precise (less variation) than estimators from the M_h and the M_{th} models after the first inspection and after the second inspection. Also, the estimators are more accurate and precise after second inspection than after the first inspection.

c) In terms of the failure rate of an estimator, M_{th}-EE failed to produce an estimate for artifact C at the first inspection. No other estimator showed any failure.

For each estimator, the relative error after the first inspection was statistically compared with the relative error after the second inspection to determine whether there was any significant improvement. For this analysis we were focused on the magnitude of the relative error and therefore overestimation and underestimation were treated equally. The results from a 2-tailed paired samples t-test show a significant improvement in the estimation accuracy for all the estimators after second inspection over the estimates after first inspection.

5.2 Selection of the Best Capture-Recapture Model

This study includes four CR models with different estimators for each model: three estimators each for M_o and M_{th} models and four estimators each for M_h and M_t models. This section analyzes the best model(s) to use in software inspections based on the estimates after the first inspection and after the second inspection. We select the best model(s) using the selection procedure (with little modification because of no outliers in our data) originally used by Briand et al., [2], explained as follows.

a) First, the best estimator for each model is selected. To choose the best estimator, a 2-tailed paired samples t-test is used to compare the relative error values for the four artifacts (to analyze the direction of improvement and its significance). For each model type, each estimator is compared against
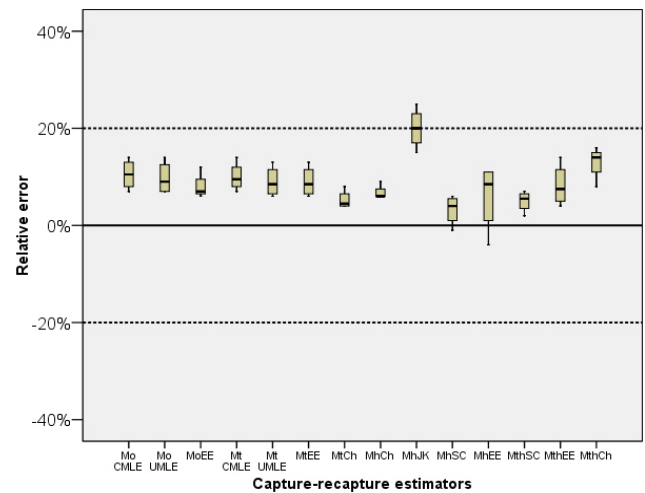

Figure 1. Relative error in estimates after first inspection

Figure 2. Relative error in estimates after second inspection

every other estimator to select the best estimator for that model. If the t-test does not show significant results in spite of the difference in their mean relative bias values, then the Wilcoxon test is performed (as this test is more powerful under certain situations and can test variability in the estimates). If there is still no statistically best estimator, the estimator with least mean relative error is chosen as the best estimator.

b) After selecting the best estimator from each model, the models are compared using 1-tailed t-test to select the best model (major source of variation). We use 1-tailed t-test to test the hypothesis that more sources of variation significantly decreases the bias in the estimate, i.e., M_{th} (with two sources of variation) is better than M_t and M_h (with one variation source), which in turn are better than M_o (with no variation).

For both statistical tests, an alpha value of 0.1 is selected. This procedure is conducted separately after the first inspection and the second inspection as shown in Figure 3 and Figure 4 respectively. In these figures, the nodes represent the best estimator selected from each model and the lines represent the relationship between models with the arrow pointing towards the better model. The p-value represents the statistical significance of the improvement (double bold lines indicate significant improvement).

5.2.1 Selection of the Best Capture-Recapture Model after First Inspection

For the M_o model, the results from the 2-tailed t-test show that CMLE is the best estimator. For the M_t, model, the results show that Chao is the best estimator. For the M_h model, the results from the t-test and Wilcoxon tests do not show any difference among the EE, Ch, and JK estimators. We selected the JK estimator because it had the smallest mean relative bias. Finally, Chao was selected the best estimator for M_{th} type model based on the results of the t-tests.
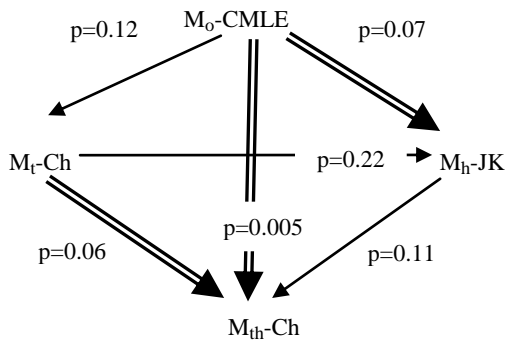
model is better than the M_t model, there is no significant difference between the M_h and M_t type models. Based on the number of arrows pointing towards a model, the M_{th} is chosen as the best model for the first inspection.

5.2.2 Selection of the Best Capture-Recapture Model after Two Inspections

For the M_o model, the results showed that EE and UMLE are the best estimators, with no significant difference among them. We selected M_0-EE as the best estimator due to its smallest mean relative error. For the M_t model, the results from the t-test show that Chao is the best estimator. For the M_h model, the results show that SC and Ch are the best estimators, with no significant difference among them. We selected M_h-SC as the best estimator due to its smallest mean relative bias. Again, the M_{th} model results showed that SC and EE are the best estimators, with no significant difference among them. We selected M_{th}-SC as the best estimator because of its smallest mean relative bias.

Figure 4, then, shows the results of the selection process for the best model after the second inspection. The results show that the M_t, M_h, and M_{th} models show significant improvement over the M_o model, which is expected (models with any source of variations are expected to be better than model with no source of variation [15]). However, both the M_t and M_h models show a non-significant improvement over the M_{th} model, which was unexpected. Based on the results in Figure 4, the model with two sources of variation (M_{th} model) is no better than models with one source of variation (M_h and M_t models). On the contrary, the M_t and M_h models showed an improvement over M_{th} model, even though the improvement is not significant. This result also contradicts the earlier finding that the M_{th} model is better than other models when there are a large number of defects and inspectors [2]. Furthermore, there is no significant improvement between the M_t and M_h models. Based on the number of arrows pointing towards a model, the M_h model is chosen as the best model after the second inspection.
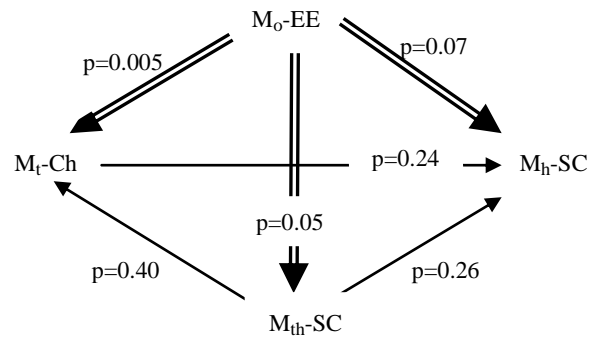

Figure 3. Best model (s) after first inspection

Figure 4. Best model (s) after second inspection

Figure 3, then, shows the results of the process for selecting the best model after the first inspection. The results from the 1-tailed t-test show that the M_t, M_h, and M_{th} models are an improvement over the M_o model, but this improvement is only significant for the M_h and M_{th} models. Therefore, the model M_t is not a significant improvement over M_o model. Furthermore, the M_{th} model is also a significant improvement over the M_t model, but not a significant improvement over the M_h model. Although M_h

Combining the results from Figures 3 and 4, we can make following observations:

a) Model with two sources of variation do not always show an improvement over models with one source of variation, but always show significant improvement over a model with no sources of variation.

b) The models Mh and Mth are a significant improvement over M_0 model, whereas the model M_t does not always significantly improves over the M_o model.

c) Neither of the models with one source of variation (i.e. M_t and M_h) is significantly better than the other.
d) There are no general trends in the selection of the best estimator for each model type (except Chao estimator for the M_t model).

5.3 Using the CR Models to Manage the Software Inspections

A major thrust of this study is to investigate whether project managers can trust CR estimates to manage inspection of software artifacts in real-time. Accordingly, this section analyzes the ability of the CR estimators to provide insight into the quality of the software artifacts. To accurately manage the inspection process, the decision about whether to re-inspect an artifact based on the CR estimates after the first and the second inspection is also evaluated. Since we do not know the actual number of defects in the artifacts, the inspectors' subjective estimates are also analyzed to gain more insights into the results.

The estimates of defects remaining in an artifact help in deciding when to stop the inspection process. We first compare the estimated remaining defects after first inspection and then, after the second inspection with the 20% threshold to make a decision on the need for re-inspection (i.e., if the defects remaining are greater than 20%, a re-inspection is needed).

As in actual development, CR models are used to estimate the remaining defects after an inspection using data only from that inspection. Accordingly, we estimate the remaining defects after first inspection without using the defect information from second inspection. The percentage of remaining defects after an inspection is calculated for each estimator as:

Relative estimate of remaining defects = (Estimated total defects - defects captured during an inspection) / Estimated total defects

For example, for artifact A and M_o-CMLE estimator, the number of defects remaining after first inspection are;

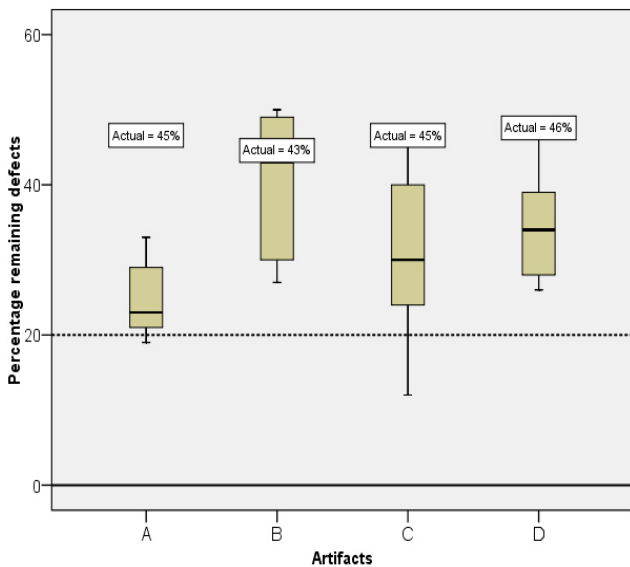$$Remaining\ defects = \frac{39(estimated\ defects) - 30\ (found)}{39\ (estimate\ total\ defects)}$$
$$= .23\ (i.e., 23\%\ of\ defects\ remain)$$

The estimates of the remaining defects are done relative to each estimator, and the estimates from all the 14 estimators are used to compute the median and range of the percentage of remaining defects for each artifact. Figure 5 compares the percentage of remaining defects for all artifacts after the first inspection (the dotted line shows the 20% threshold). Figure 5 also shows the actual percentage of additional defects found during the second inspection for each artifact to give an indication of the accuracy of the estimates after the first inspection.

After the first inspection, the median estimated remaining defects for all the artifacts is greater than 20%, while some estimators estimated 40% or more remaining defects for artifacts B, C, and D. The estimates indicate the need for a re-inspection of all artifacts. The actual data from the second inspection showing the percentage of the additional defects found during re-inspection (A- 45%, B- 43%, C- 45%, D- 46%) supports this decision. During the original study, the CR models were not used, so the only data which was available to decide on a re-inspection was the subjective opinions of the inspectors. In this case, the subjective opinion of developers supported the recommendation of the CR estimators. The inspectors for each artifacts felt that there were defects remaining after first inspection and so a re-inspection was performed. Similarly, the number of remaining defects after the second inspection is estimated relative to each estimator using the defect data from both inspections.

For example, for artifact B and M_t-Ch estimator:

$$Remaining\ defects = \frac{91(estimated\ defects) - 81(found)}{91\ (estimate\ total\ defects)}$$
$$= .11\ (i.e., 11\%\ of\ defects\ remain)$$

The percentage of estimated remaining defects after the second inspection for each artifact is shown in Figure 6. After the second inspection, the median estimate of remaining defects as well as extreme outliers for all the artifacts never exceeds the 20%

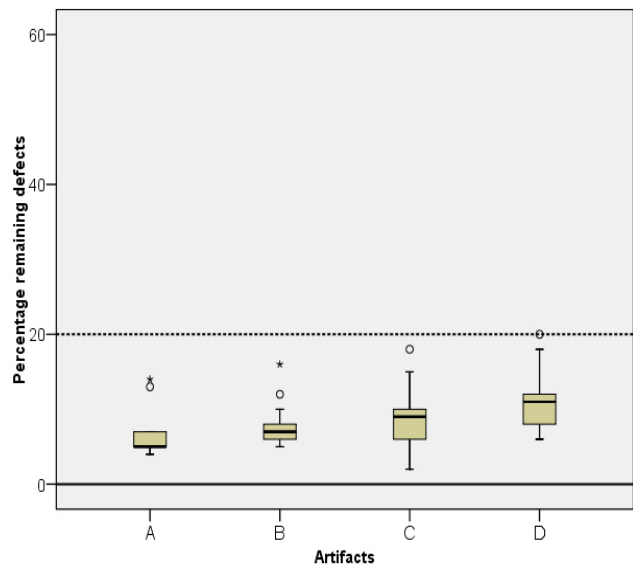**Figure 5. Estimated remaining defects after the first inspection**

Figure 6. Estimated remaining defects after the second inspection

threshold. There is some variation in the estimates for artifacts C and D, but none of the estimate exceeds 20%. The results indicate that there is no need for further re-inspections for any of the artifacts; and the inspection process should be stopped.

The inspectors' subjective opinion regarding the remaining defects after the second inspection (which was all that was available during the original study) supported the recommendation of the CR estimates. The inspectors agreed that they had located all the defects present in the artifact during second inspection, ruling out any need of further inspection. So, the inspection process was stopped.

Combining the results from Figure 5 and 6, the CR estimates is helpful to managers for deciding on the need of re-inspection under realistic inspection conditions.

6. THREATS TO VALIDITY

In this study, there were some threats to validity that were addressed. The artifacts used in this study are real software artifacts that were later used to guide implementation. The defects were naturally occurring and inserted while developing the artifacts rather than artificially seeded. The subjects were provided an equal amount of time to perform the first and second inspection, thereby avoiding any bias.

However, there were some threats to validity that were not addressed. First, the actual number of defects present in each document is not known and might actually be higher than the assumed defect count (i.e., the total number of defects found after two inspections). This threat especially affects the evaluation of the prediction after the first inspection. A second threat was the artifacts used in this study were developed by student teams in a senior-level capstone course, and it may not be a representative of industrial strength requirement documents. Also, the nature of faults committed by students during development can differ from the faults made by software professionals.

7. DISCUSSION OF RESULTS

This section discusses the major findings and recommendations about the CR models and estimators for achieving reliable estimates. The major findings from this study are compared with the earlier findings from software engineering and biology.

7.1 Summary of Major Findings

Quality of estimates: As expected, the number of defects influences the quality of estimates. The estimates are highly negatively biased after first inspection and become positively biased after second inspection as more defects are found. The improvement is statistically significant for all the estimators. Considering the fact that there might be more defects remaining after second inspection, that were not included in the analysis, the estimates may actually be more negatively biased after first inspection and less positively biased after second inspection. In addition, the precision and the failure rates of the estimators also improve after the second inspection. However, contrary to earlier findings, M_h-JK is not the best estimator. In this study, M_h-JK overestimated compared with other estimators after both inspection cycles. This overestimation was unsatisfactory after the second inspection. Although, because we do not know how many defects actually remain after the second inspection, it is possible that M_h-JK has not overestimated.

The best capture-recapture model: We tested the hypothesis that: M_{th} model with two sources of variation (varying inspector abilities and varying defect detection probabilities) is significantly better than models M_t (varying inspector abilities) and M_h (varying defect detection probability) with one source of variation, which in turn are significantly better than M_o model with no variation. Using the estimates from the first inspection alone, the result did not followed this trend since the M_{th} model did not show a significant improvement over M_h model, and the M_t model did not show a significant improvement over M_o model. The results are even more unexpected after the second inspection, as there is no significantly better model between the models with one source of variation (M_t and M_h) and the model with two sources of variation (M_{th}). In summary, M_{th} is the best model after the first inspection whereas M_h is the best model after the second inspection. Therefore, defect detection probability is always a source of variation, while the inclusion of varying inspector ability does not always improve the estimation. However, there is no general trend regarding the best model under all cases. Lastly, no particular estimator for each model type outshines other estimators all the time.

Determining quality of a software artifact: The results show that the CR estimators can help managers accurately decide on the need for re-inspection of an artifact. Some estimators can underestimate the actual defect count and hence, the remaining defects in an artifact. It is therefore recommended that all the estimators are used. Then analyze the median and variability in the estimates of remaining defects. If any data point exceeds the 20% mark, then re-inspection should be considered. The results also show that the subjective estimates are similar to objective estimates if the inspectors are same people who developed the requirements because they can provide much better assessment. It is not clear how the subjective estimates would differ if the inspectors were not involved in the artifact development. This is a future research issue that must be investigated.

7.2 Comparison with Previous Findings in Software Engineering and Biology

Table 4 compares the major findings from this study with earlier research findings from software engineering and biology and wildlife research. The findings from this study support some of the previous findings, while they contradict some others, and provide some additional insights.

The findings from software engineering and biology that are consistent with our findings are: 1) CR estimates improve as more defect data is fed to them, 2) The M_{th} model does not improve significantly over M_h model, and 3) The M_h-JK estimator overestimates with less overlap of recaptured defects.

Some of the findings in this paper that contradicts the earlier findings are: 1) There is no significant difference between M_h and M_t models, 2) The M_t model is not always a significant improvement over M_o model, 3) There is no best estimator for all the CR model all the times, and 4) It is not always true that models with more sources of variations are significantly better.

Some of the findings reported in this paper were new. The jackknife (JK) estimator always overestimates in comparison to all the other 13 estimators used in this study. Also, a relatively new result showed the ability of the CR models to accurately decide the need for re-inspection of software artifacts in real-time development. The subjective opinion of the inspectors (also used

Table 4. Comparison of Findings

S. No	Our Study	Software Engineering	Biology and Wildlife
1.	CR estimators highly underestimate the defect count after the first inspection, and the estimates are significantly more accurate and precise after the second inspection.	The CR estimators underestimate the actual number of defects, and the estimates improve with more defects, and inspectors [16, 20, 21]	All models generally underestimate but estimates improve with more trapping occasions and animals, Failure rate is high only for few inspectors [15, 12, 22]
2.	M_h-JK overestimates after each inspection compared with the other estimators and the overestimation increases with decreasing defect overlap.	Studies show that M_h-JK overestimates if the overlap of defects among inspectors is small [2, 16, 20]	Mh- JK severely overestimates in case of few trappings, but provide good estimates if the overlap of animals caught at different trappings is large [6, 22]
3.	M_{th} is the best model after the first inspection whereas M_h is the best model after the second inspection; the defect capture probability is one definite source of variation	A large number of studies indicate that the M_h is the best CR model, and defect capture probability is a major variation source [2, 13, 16]	M_h show significant improvement over M_t models, while M_{th} models do not show improvement over M_h models [12, 22]
4.	We contradict that M_h-JK is the best estimators as it do not always produce the best results	Studies show that M_h-JK is the best estimator with four or more inspectors [2, 13, 16]	Mh-JK produces good estimate especially if many animals are recaptured a lot of times [3, 15, 22].
5.	There is no general trend in the best estimator for each model. Different estimators for each model type are best at first and second inspections.	UMLE is the best estimator for M_o model, Ch is best for M_t and M_{th} model, Ch and JK are best for M_h model [2], SC for Mh and Mth are best estimators [20].	MLE estimators for M_o and M_t type models produce highly inaccurate estimates as compared to M_h models [15]
6.	Results from this study do confirm that the M_{th} model is not always a significant improvement over M_t and M_h models, and is always a significant improvement over M_o model. We reject that M_t is always a significant improvement over M_o, and that the M_h model is significantly better than the M_t model.	A general trend is that the more sources of variations, the better the model [13, 16], Model M_h is more usable than Model Mt, while M_{th} does not always significantly improves over M_h [2].	A general trend is that the more sources of variations, the better the model, but M_{th} does not always significantly improves over M_h [15, 22]
7.	CR estimators can provide accurate information about whether or not to stop the inspection process.	CR estimators are not always accurate in deciding whether to re-inspect [11].	Not Applicable
8.	The inspectors' subjective estimates match the CR estimates when the inspectors are same people as the developers.	Some studies show similarities between the subjective and objective estimates, while others show a significant difference [10, 18].	No Result

in this study) matched with the CR estimates when the inspectors are same as developers.

7.2 Relevance to Software Organizations

The CR models have been widely used in the academic context, but hardly in industrial settings. Our earlier research efforts analyzed the inspection defect data from Microsoft Corporation to determine the minimum number of inspectors required to achieve different levels of estimate accuracy [20]. This information benefits project managers to plan and manage the inspection process. Similarly, the results in this paper encourage organizations to use the CR models in order to manage software inspections to achieve their desired software quality standards. The results in this paper add some more support to the view that the CR models are usable by mangers to make correct re-inspection decisions during software development practice. Project mangers can use the results about the CR models and estimators to make an informed decision on the need for re-

inspection. Indeed, a lot more research is needed to back these results. The major future research issues are addressed in the next section.

8. CONCLUSION AND FUTURE WORK

Based on the results provided in this paper, project managers and software developers can use the CR models and estimators in software organizations to manage software inspections and achieve the desired artifact quality. We have evaluated the estimators based on a +/-20% threshold. Project managers can interpret these results using the specific quality criterion of their organization. In addition, the cost-effectiveness of a re-inspection should be considered in conjunction with the estimates of remaining defects.

The results showed that the number of defects and the overlap of defects among inspectors influence the performance of estimators. Our future work in this area includes analyzing the effect of the

number of defects and the overlap of defects using this data set by varying the number of defects and the number of inspectors. An optimal number of defects and inspectors would provide more information to project managers and software developers in conducting effective software inspections. We also want to analyze the influence of the inspection technique effectiveness and other important variables on the performance of the CR estimators.

9. ACKNOWLEDGEMENTS

10. REFERENCES

[1] Ackerman, A., Buchwald, L., and Lewski, F., "Software Inspections: An Effective Verification Process*." IEEE Software*, 1989. **6**(3): 31-36.

[2] Briand, L.C., Emam, K.E., Freimut, B.G., and Laitenberger, O., "A Comprehensive Evaluation of Capture Recapture Models for Estimating Software Defect Content*." IEEE Transactions on Software Engineering*, 2000. **26**(6): 518-539.

[3] Burnham, K.P. and Overtom, W.S., "Estimation of the Size of a Closed Population When Capture Probabilities Vary among Animals." *Biometrics*, 1978. 65:625-633.

[4] Chao, A., "Estimation the population Size for Capture-Recapture Data with Unequal Catchability*." Biometrics*, 1987. **43**(4): 783-791.

[5] Chao, A., "Estimating Animal Abundance with Capture Frequency Data*." Journal of Wildlife Management*, 1988. **52**(2): 295-300

[6] Chao, A. and Yeng, H.C., *Program CARE-2 (for Capture-Recapture Part.2), http://chao.stat.nthu.edu.tw*

[7] Darroch, J.N., "The Multiple-Recapture Conensus 1: Estimation of a Closed Population." *Biometrika*, 1958. **45**: 343-359.

[8] Eick, S., Loader, C., Long, M., Votta, L., and Weil, S.V. "Estimating Software Fault Content Before Coding". In *Proceedings of the 14th International Conference on Software Engineering*. 1992. Melbourne, Australia: ACM Press: 59-65.

[9] Eick, S., Loader, C., Weil, S.V., and Votta, L. "How Many Errors Remain in a Software Design after Inspection". In *Proceedings of the 25th Symposium on the Interface*. 1993.

[10] El-Emam, K., Laitenberger, O., and Harbrich, T., "The Application of Subjective Estimates of Effectiveness to Controlling Software Inspections *" Journal of Systems and Software*, 2000. **54**(2): 119-136.

[11] El-Emam, K. and Laitenberger, O., "Evaluating Capture-Recapture Models with Two Inspectors*." IEEE Transactions on Software Engineering*, 2001. **27**(9): 851-864

[12] Lee, S.M. and Chao, A., "Estimating Population Size via Sample Coverage for Closed Capture-Recapture Models*." Biometrics*, 1994. **50**: 88-97.

[13] Miller, J., "Estimating the Number of Remaining Defects after Inspection*." Software Testing, Verification and Reliability*, 1999. **9**(3): 167-189.

[14] Musa, J., Iannion, A., Okumoto, O., "*Software Reliability: Measurement, Predicition, Application*," McGraw-Hill, 1987

[15] Otis, D., Burnham, K., White, G., and Anderson, D., "Statistical Inference from Capture Data on Closed Animal Population*." Wildlife Monograph*, 1978. **64**: 1-135.

[16] Petersson, H., Thelin, T., Runeson, P., Wohlin, C. "Capture-Recapture in Software Inspections after 10 Years Research-Theory, Evaluation, and Application." *The Journal of Systems and Software*, 72(2):249-264.

[17] Runeson, P. and Wohlin, C., "An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections*." Empirical Software Engineering: An International Journal*, 1998. **3**(4): 381-406.

[18] Thelin, T., Petersson, P., and Runeson, P., "Confidence Intervals for Capture-Recapture Estimations in Software Inspections*." Journal of Information and Software Technology*, 2002. **44**(12): 683-702.

[19] Walia, G.S. and Carver, J. C., Philip, T., "Requirement Error Abstraction and Classiifcation: An Empirical Study," In *Proceedings of 5th International Symposium on Empirical Software Engineering*, Rio de Janerio, 2006, pp. 336-345.

[20] Walia, G., Carver, J., and Nagappan, N. "The Effect of the Number of Inspectors on the Defect Estimates Produced by Capture-Recapture Models." To appear in the *Proceedings of the 30th International Conference in Software Engineering*. May 10-18, 2008. Leipzig, Germany.

[21] Weil, S.V. and Votta, L., "Assessing Software Designs Using Capture-Recapture Methoda*." IEEE Transactions on Software Engineering*, 1993. **19**(11): 1045-1054.

[22] White, G.C., Anderson, D.R., Burnham, K.p., and Otis, D.l., *Capture-Recapture and Removal Methods for Sampling Closed Populations,* Los Alamos National Laboratory*, 1982.*

[23] Wohlin, C., Runeson, P., and Brantestam, J., "An Experimental Evaluation of Capture-Recapture in Software Inspections*." Software Testing, Verification and Reliability*, 1995. **5**(4): 213-232.

[24] Wohlin, C. and Runeson, P. "Defect Content Estimation from Review Data". In *Proceedings of the 20th International Conference on Software Engineering*. 1998. Kyoto, Japan: IEEE Computer Society Press: 400-409.

[25] Yip, P.S.F., "A Martingale Estimating Equation for a Capture-Recapture Experiment in Discrete Time*." Biometrics*, 1991. **47**: 1081-1088.