P. Runeson and C. Wohlin, "An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections", Empirical Software Engineering: An International Journal, Vol. 3, No. 4, pp. 381-406, 1998.

# An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections

**Per Runeson and Claes Wohlin**
**Dept. of Communication Systems, Lund University**
**P.O. Box 118, SE-221 00 Lund**
**(perr, claesw)@tts.lth.se**

## Abstract

In order to improve the efficiency of inspections, quantitative data on defect content have to be the basis for decisions in the inspection process. An experience-based capture-recapture method is proposed, which overcomes some problems with the basic pre-requisites of the original method. A C-code inspection experiment is conducted to evaluate the enhanced method and its applicability to software code inspections. It is concluded that the experience-based estimation procedure gives significantly better estimates than the maximum-likelihood method and the estimates are not very sensitive to changes in the inspection data.

## Keywords

Software inspection, Capture-Recapture, Defect content estimation, Experience-based estimation, Experiment.

## 1. Introduction

Inspections are accepted widely in the software engineering community as efficient contributors to improved software quality and reduced costs. The efficiency of inspections are seldom questioned anymore. Typical results according to Gilb (1993) show that inspections' direct savings pay back the invested time 2-6 times. However for a specific inspection there is a need for quantitative methods to analyse the outcome of an inspection. The data generally available after an inspection are the number of defects identified and removed. However, these defects do not cause any problem when being absent. It is more important to know how many defects that remain in the software artifact, but this information is generally not available.

If the information on remaining defects was available, it could be applied to control the development process, in order to utilize the development resources in the most cost-effective manner. Based on the estimate of remaining defects, informed decisions could be taken whether to accept the quality of the artifact or not. If the estimated number of defects is below a given threshold, the artifact can be released for the next phase. Otherwise, it has to be improved by, for example, a re-inspection.

The capture-recapture method is a simple, but useful approach to quantify remaining defects after inspection. The application of capture-recapture to software is proposed by Eick et al. (1992); further studies and improvements are presented by Vander Wiel and Votta (1993), Wohlin et al. (1995), Briand et al. (1997) and Ebrahimi (1997). Related methods are proposed by Wohlin and Runeson (1998) and evaluated by Briand et al. (1998). The principles behind the capture-recapture method are intuitively right, however there are some basic problems in the preconditions for the estimation models applied. The $M_t$ model[1] with maximum-likelihood (M-L) estimator is derived from the assumptions that different defects have the same probability of being revealed, while the reviewers are assumed to have different probabilities for finding the defects. The $M_h$ model with jackknife estimator assumes the opposite, namely that the defects have different probabilities for being found, while all reviewers are assumed to have the same probability for finding a specific defect.

Neither of the two assumptions is sufficient. Different defects do not have the same detection probabilities and different reviewers do not have the same probability for finding a specific defect, depending on for example their background and skill.

## 2. Capture-recapture estimations

### 2.1 Introduction

In (Vander Wiel and Votta, 1993) it is concluded that the capture-recapture with maximum-likelihood estimates consequently gives underestimated values. To address the above outlined problem, a filtering approach is proposed in (Wohlin et al., 1995). The proposed method is evaluated in an inspection experiment in which a text document was inspected.

The principle behind the filtering approach is that the defects found are divided into two classes based on a filter. One type of filter ensures that the faults found by only one reviewer are collected in one class. Then a multiplicative factor is applied to that class and the capture-recapture method with M-L estimates is applied to the other class of defects.

The underlying assumption for this approach is that the $M_t$ model with M-L estimate takes into account that different reviewers have different probabilities for finding a specific defect, while all defects are assumed to have the same detection probability. In the filtering approach we assume that the defects found by few reviewers have a lower detection probability and those found by many reviewers have a higher detection probability. By handling these classes separately, the drawbacks of the M-L estimates are reduced.

The filtering approach is briefly introduced in Section 2.3 below and elaborated in depth in (Wohlin et al., 1995). The results are promising and shows improved estimates

1. $M_t$ means model with variation by time response and $M_h$ means model with variation by heterogeneity (Otis et al. 1978; White et al. 1982). In this paper we refer to the models by the name of the estimators used, maximum-likelihood (M-L) and jackknife respectively.

of the defect content. In this paper, the multiplicative factor in the filter-based estimation technique is enhanced by introducing an experience-based multiplicative factor. The factor takes different reviewer profiles into account and helps organisations continuously improve their estimates.

In order to evaluate the application of capture-recapture and the experience-based multiplicative factor to code inspections, a C-code inspection experiment is conducted with eight reviewers reading five different programs with known defects. In this paper we report the experiment and evaluate the application of capture-recapture estimations applying the filtering technique with experience-based factors on software code inspections.

There is nothing in the model principles that hinders applying capture-recapture models to inspection of any type of software artifacts. Inspections of requirements and design documents and capture-recapture have been evaluated by Eick et al. (1992) and Briand et al. (1998). In the experiment presented here we apply capture-recapture to code inspections. In practice, code inspections are carried out by fewer people and usually after compilation. These constraints imply that fewer defects are found and the overlap between reviewers is smaller, making it harder to get stable estimates from the capture-recapture estimators.

In Section 2.2, the maximum-likelihood estimator is presented and in Section 2.3 the filter approach is elaborated.

## 2.2 Maximum-Likelihood estimator

The M-L estimate can be derived by numerically maximizing the following equation over $N \geq n$, see (Eick et al., 1992):

$$L(N) = \log\binom{N}{n} + \sum_{i=1}^{m} n_j \log n_j - Nm\log N + \sum_{i=1}^{m} (N - n_j)\log(N - n_j) \qquad (1)$$

where the following notations are used:

$N$ – estimated initial number of defects

$n$ – total number of unique defects found

$n_j$ – number of defects found by reviewer $j$

$m$ – number of reviewers

More details on the maximum-likelihood and other estimators can be found in (Otis et al., 1978; White et al., 1982).

## 2.3 Filter with multiplicative factor

In order to reduce the effects of the varying detection probabilities for different defects, the defects are sorted into one of two classes via a filter. In the experiment reported in (Wohlin et al., 1995) two types of filters are evaluated: 1) a filter which divides the defects into classes based on the percentage of the reviewers that found the defect, and 2) a filter which puts all defects found by one reviewer in one class. In this experiment we apply the latter since it is more suited in inspections with fewer participants. The filter applied to this experiment ensures that the defects found by a single reviewer belongs to class 1. The rest of the defects belong to class 2.

The M-L estimate requires that there is an overlap between reviewers. It fails to estimate the number of defects in class 1, since there is no overlap. Instead a multiplicative factor (*f*) is applied. The multiplicative factor represents the number of defects in the class of defects, related to the ones which are actually found by a single reviewer. The M-L estimate is applied on defects in class 2.

The experience-based factor is defined as the average of the outcome of the *k* last inspections:

$$f = \frac{1}{k}\sum_{i=1}^{\kappa}\frac{N_i - \hat{N}_{\mathrm{Cl}\,2,\,i}}{n_{\mathrm{Cl}\,1,\,i}} \qquad (2)$$

where the following notations are used:

$N$ – actual number of initial defects

$\hat{N}$ – estimated number of initial defects

$n$ – total number of unique defects found

Cl 1 – class 1, i.e. defects found by a single reviewer

Cl 2 – class 2, i.e. defects found by more than one reviewer

The window length, *k*, has to be chosen depending on the available reviewers' experience, the stability of the inspection environment etc. In this case, we apply a *k* number of 3. It is selected as a trade-off between stability and flexibility in the estimates. On longer data runs, the window is to be tuned, keeping it as short as possible to remain sensitive to environmental and team composition factors versus keeping it long enough to avoid statistical variations.

To make the method simpler, it is based on the assumption that the estimates of the number of class 2 defects are correct ($\hat{N}_{\mathrm{Cl}\,2}$). Hence only the class 1 estimates have to be adjusted by the experience-based factor.

The filter estimates are derived by applying the following method:

1. Divide the data into two classes. The defects found by a single reviewer belong to class 1 and the defects found by more than one reviewer belong to class 2.
2. Apply the maximum-likelihood estimator to the defects in class 2 ($\hat{N}_{\mathrm{Cl}\,2}$).
3. Multiply the number of defects in class 1 with the experience based factor ($\hat{N}_{\mathrm{Cl}\,1} = n_{\mathrm{Cl}\,1} * f$).
4. Add the two estimates to obtain an estimate of the total number of defects ($\hat{N} = \hat{N}_{\mathrm{Cl}\,1} + \hat{N}_{\mathrm{Cl}\,2}$)
5. Recalculate the experience-based factor with the new information, see equation (2).

In the experiment presented below, this estimation procedure is evaluated against the pure M-L estimation procedure as presented in Section 2.2.

# 3. Inspection experiment

## 3.1 Experimental hypothesis

The hypothesis for the experiment is that the filtering method with defects found by a single reviewer in one class, as presented in (Wohlin et al., 1995) and an experience-based multiplicative factor is applicable to software code inspections as well. We expect the experience-based method to *give better estimates*, i. e. 1) estimates closer to the real value, 2) with less variance, and 3) less sensitive to variations in the inspection team, than the capture-recapture method with only M-L estimates does. The hypothesis is summarized as:

- Null hypothesis: The filtering method and an experience-based multiplicative factor, *does not give better estimates* than the capture-recapture method with only M-L estimates.
- Alternative hypothesis: The filtering method and an experience-based multiplicative factor, *gives better estimates* than the capture-recapture method with only M-L estimates.

The hypothesis is divided into three sub-hypotheses as defined by items 1), 2) and 3) above and it is tested by conducting an experiment with C-code inspections.

## 3.2 Inspection objects

The software programs to be inspected are defined in (Humphrey, 1995) and developed according to the Personal Software Process (PSP). The programming language used is C. Programs 3A, 4A, 5A, 6A and 7A according to the PSP course scheme are selected for inspections. Programs 1A and 2A were not selected due to risks for initialization effects and a total number of five inspection objects were selected in the experiment as a trade-off between reasonable workload for the participants and a sufficient number of inspections for the experiment.

The programs solve small data structure, statistics and numerical problems. The size of the final programs are between 85 and 304 lines of C-code, according to the defined code counting standard. Some of the code in the programs developed later, is reused from earlier developed programs. The reuse relations are presented in Figure 1 and more detailed information is shown in Table 1. The reviewers knew the reuse relations, and those who reviewed software parts which they already had reviewed before, could thus use this information to focus on non-reused parts.

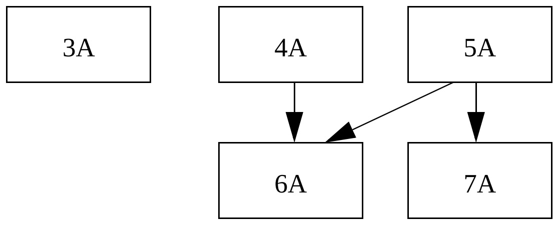

**FIGURE 1. Reuse between programs**

During the PSP development, a baseline of the software programs was frozen before any compile, inspection or test. The programs are further developed in the PSP development and thus the defects are identified, which are used as a reference to the inspection experiment. The baseline “before compile and inspection” programs are used as inspection objects in this experiment. The number of defects in the programs and their size are shown in Table 1.

**TABLE 1. Defect and size data**

| **Program** | **3A** | **4A** | **5A** | **6A** | **7A** | **Total** |
|---|---|---|---|---|---|---|
| Defects | 22 | 16 | 16 | 35 | 20 | 109 |
| LOC[a] | 190 | 113 | 85 | 304 | 208 | 900 |
| LOC $_{\text{reused}}$ [b] | 0 | 0 | 0 | 167 | 162 | 329 |

a. These figures include new and reused code.

b. Humphrey differs between library reuse and base program reuse (Humphrey, 1995). LOC $_{\text{reused}}$ include both reuse categories and are calculated as *Base–Deleted–Modified+Reused* according to the definitions by Humphrey.

The defects are hence not seeded in the code, but are real defects introduced during software development. Functional defects as well as cosmetic ones, like misspelled comments, are counted as defects. The majority of the defects are of syntax type, and the second largest class of defects are of functional type, such as logic, pointers and loops. The defects are classified according to the scheme presented in the PSP context (Humphrey, 1995), see Table 2, which is based on (Chillarege et al., 1992). Defect type data are presented in conjunction with the defect data in the annex, Table 12 to Table 16.

**TABLE 2. Defect classification scheme**

| **Class** | **Description** |
|---|---|
| 10 | Documentation |
| 20 | Syntax |
| 30 | Build, Package |
| 40 | Assignment |
| 50 | Interface |
| 60 | Checking |
| 70 | Data |
| 80 | Function |
| 90 | System |
| 100 | Environment |

## 3.3 Experiment participants

The inspection experiment was conducted by eight reviewers with various industrial and academical background. Four reviewers were PhD students (B, D, E and G), three were software engineering seniors (C, F and H) and one was a recently graduated software engineer (A). Two of the reviewers (C and F) took part also in the former text inspection experiment (Wohlin et al., 1995) but the current experiment is totally independent of the inspection objects in that study. The reviewers were categorized according to a reviewer profile scheme. They rated their experience on a scale from 1 to 4, where 1 means no knowledge or experience while 4 means more than three years experience. The profile is presented in Table 3. The table also shows the defect detection yield from the experiment, calculated as number of defects found by the reviewer, divided by the number of defects in the reviewed documents. Finally, the time spent by each reviewer is listed.

**TABLE 3. Reviewer profile for reviewers A-H**

| **Area** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** |
|---|---|---|---|---|---|---|---|---|
| SE state of the art knowledge | 3 | 3 | 4 | 3 | 4 | 4 | 3 | 4 |
| SE practice | 2 | 2 | 4 | 2 | 2 | 4 | 2 | 4 |
| C coding experience | 1 | 3 | 3 | 3 | 2 | 4 | 2 | 2 |
| Problem domain experience | 1 | 3 | 2 | 3 | 3 | 2 | 3 | 2 |
| Inspection experience (general) | 3 | 2 | 3 | 2 | 3 | 3 | 2 | 4 |
| Inspection experience (C-code) | 1 | 2 | 1 | 2 | 2 | 1 | 2 | 1 |
| Defect detection yield | 0.535 | 0.276 | 0.123 | 0.411 | 0.323 | 0.204 | 0.366 | 0.278 |
| Time spent (minutes) | 217 | 170 | 117 | 199 | 88 | 200 | 198 | 153 |

Three types of reviewer profiles can be identified among the reviewers (experience levels from the profile given in parentheses):

- Seniors within SE (4), but less familiar within problem domain (2) and C code inspection (1) - reviewers C, F and H.
- Less experienced within SE (2–3), but more familiar within the problem domain (3) and C code inspection (2) - reviewers B, D, E and G.
- Less senior within SE (2–3) as well as problem domain (1) and C code inspection (1) - reviewer A.

Correlation analysis is conducted between detection yields versus the characteristics of the reviewers and the time spent by each reviewer. An analysis is conducted with reviewer A removed as well, since this reviewer has low experience rating but high detection yield. The correlation analysis is presented in Table 4.

**TABLE 4. Correlation between reviewer profile and defect detection yield**

| | | **All reviewers** | | **Reviewer A removed** | |
|---|---|---|---|---|---|
| | | **r** | α | **r** | α |
| 1. | SE state of the art knowledge vs yield | –0.696 | 0.055 | –0.654 | 0.118 |
| 2. | SE practice vs yield | –0.761 | 0.026 | –0.767 | 0.043 |
| 3. | C coding experience vs yield | –0.700 | 0.052 | –0.446 | 0.337 |
| 4. | Problem domain experience vs yield | –0.155 | 0.726 | 0.782 | 0.036 |
| 5. | Inspection experience (general) vs yield | –0.233 | 0.596 | –0.474 | 0.030 |
| 6. | Inspection experience (C-code) vs yield | 0.249 | 0.569 | 0.782 | 0.036 |
| 7. | Time vs yield | 0.529 | 0.188 | 0.347 | 0.470 |

The three first areas are correlated to the defect detection yield ($|r| \approx 0{,}7$, $\alpha < 0{,}06$). However, the correlation is negative, thus implying that increased experience has a negative impact on the result. One explanation to this result can be that the reviewer profile does not take into account whether the experience is fresh or some years old. The SE seniors had not worked with low level coding for some years, and the least experienced reviewer in SE in general and C coding was actively working with other coding languages. This indicates that the reviewer profile has to capture other aspects than the one currently used.

Removing reviewer A, which seems to differ from the other with respect to experience rating vs detection yield, gives slightly different results. Still the first two areas are negatively correlated to the detection yield. Now, however, the problem domain experience and C code inspection experience is positively correlated to detection yield.

It is important to notice that the varying detection yield between reviewers emphasizes the need for using estimation procedures which allow for different probabilities in finding defects, as identified in Chapter 2. This implies that the basic assumption of the jackknife estimator is violated, which assumes that all reviewers have the same detection yield. The drawbacks of jackknife estimations is confirmed by Vander Wiel and Votta (1993). Hence in the rest of this paper we are working with the M-L estimator. In later research, opposite results are shown and the jackknife estimator is found to be better than the M-L estimator (Briand et al., 1997).

## 3.4 Experiment execution

Each of the reviewers reviewed three programs out of the set of five. The programs were randomly allocated to the reviewers. Programs 3A-6A were reviewed by five reviewers each, while program 7A was reviewed by four reviewers.

The reviewers were given a package for each program containing:

- Code review process with:
  - corresponding code review checklist
  - time recording log
  - defect recording log
- C program code listing
- Requirements specification from the PSP book (Humphrey, 1995)
- Program design (textual)

The reviewers performed individual inspections. They were informed that the estimated inspection time was an hour per program, but they were free to spend more or less time if they liked to. Actually spent time varied between 88 and 217 minutes for three programs, see Table 3. The programs were inspected in consecutive order, time spent and defects found were logged and the results were collected. No common inspection meeting was held due to calendar management problems. The absence of a inspection meeting is not considered a threat to the experiment. We point to the continuing debate about the value of the inspection meeting as discussed by (Votta, 1995; Porter et al., 1995; Johnson and Tjahjono, 1998).

After the inspection, the data were collected and analysed. Before the analysis, judgement was made on which defects are true ones and which are false positives. For example some reviewers had counted multiple defects which others had counted as single ones. Furthermore, the reviewers logged suspect defects where they were not really sure whether it was correct or not. Defects are counted independently of whether they are seriously affecting functionality or are minor misspellings or violations of coding rules.

The resulting defect data are presented in the annex, Table 12 to Table 16.

## 3.5 Threats to the experiment

Setting up an experiment involving human subjects always includes risks that parameters out of experimental control affect the result. Identified threats to the validity are presented below and their impact on the result is evaluated. Threats to the internal validity are:

**Selection effects.** The reviewers were allocated randomly to the programs to be reviewed. The purpose was not to compose a specific competency profile team, but to compose a team out of the available competency. The inspection order of the programs was not random, but consecutive, which might be a threat to the experiment. However, since all reviewers did not inspect all programs, the issue is not critical. For example, program 5A is inspected as the last program by a reviewer assigned to 3A, 4A and 5A while it is inspected as the first program by a reviewer assigned to programs 5A, 6A and 7A.

**Maturation effects.** The objects were inspected in increasing order, thus there is a risk of being better skilled at the third inspection than at the first one. On the other hand, motivation decreases with time which makes the reviewers less motivated in the later inspection. The detection yields in Table 9 show examples of both effects, thus it is assumed that there is no systematic impact on the result.

**Instrumentation effects.** The reviewers were aware of that they took part in an inspection experiment, thus they might act different than they usually did. On the other hand, the experiment's purpose is to evaluate two different estimation methods. The inspections in itself are not the study object, but the data generated from the inspections.

Threats to external validity of the experiment are:

**Experimental scale.** The inspected code was written by a novice in C programming. Hence the share of syntactic defects was high, 51%, giving the risk that the results are not valid externally in an environment with another defect type distribution.

The inspected programs were small, and the results might not be scalable into larger problems. However the programs are of the magnitude of size possible to inspect in one step (Fagan, 1986), so from that perspective the results are applicable on parts of larger programs as well.

**Subject generalizability.** Some of the reviewers (B, D, E and G) had better domain knowledge, in that they had conducted the PSP course a few months before the experiment took place. There is a risk that this affects the results. However, reviewer A, who had not conducted the PSP course before, also had higher detection yield than the experienced seniors, and there is no significant correlation between problem domain experience and detection yield, see Table 4, at least when considering all reviewers.

**Subject representativeness.** The reviewers were selected among PhD students within software engineering and software engineering seniors in industry. The reviewers represent detailed C programming knowledge as well as general software engineering experience, which is representative of available experience in industry, but the share of the latter group in this experiment is probably higher than in a code inspection team in industry.

## 4. Data analysis

### 4.1 General

In the analysis of the experiment data, two different estimation techniques are applied to estimate the number of defects in the inspected code. First an M-L estimation is applied (Eick et al., 1992). Secondly the filter approach with an M-L estimation combined with an experience-based factor is used (Wohlin et al., 1995). A sensitivity analysis of both methods is conducted where the data set is reduced by withdrawing one reviewer, and the effect on the estimates are studied.

The goodness of the results is evaluated using the estimate's numerical and percentage error compared to the actual value. The percentage error is defined as:

$$\%\text{error} = \frac{100(\hat{N} - N)}{N} \tag{3}$$

where:

$N$ – actual number of initial defects

$\hat{N}$ – estimated number of initial defects

The percentage error as well as the absolute percentage error are analysed.

It shall be noted that the limited number of defects in the code makes the estimates sensitive to single defects as well as to the fact that the number of defects is discrete. A single defect contributes to the estimate with between 2.8 and 6.3 percent for programs with 35 and 16 defects respectively.

It can be argued that the syntax errors should not be included in the study since industry practice is that inspections are performed after compile. The analyses presented below are conducted without the syntax defects (type 20) as well. The results of these analyses are consistent with the presented ones, but the estimation errors have a larger variation, since the models are more sensitive to the change of a single data point when we have fewer defects.

## 4.2 Maximum-Likelihood

Applying the M-L estimator on the experiment data gives results between the correct estimate and 27% overestimate, see Table 5. The mean value of the percentage error in this study is 10.5% and standard deviation is 10.7%.

**TABLE 5. Maximum-Likelihood estimates**

| **Program** | $n$ | $\hat{N}$ | $N$ | $\hat{N}-N$ | **%error** |
|---|---|---|---|---|---|
| 3A | 18 | 28 | 22 | +6 | +27% |
| 4A | 15 | 16 | 16 | 0 | 0% |
| 5A | 15 | 17 | 16 | +1 | +6% |
| 6A | 33 | 39 | 35 | +4 | +11% |
| 7A | 20 | 25 | 20 | +5 | +25% |
| | | | | Mean[a] | +10.5% |
| | | | | St.dev. | 10.7% |

a. The mean and standard deviation values are based only on programs 4A-7A since these figures are used for comparison with the filtering technique below.

The percentage errors for the analysis without syntax defects are 0%, 29%, 50%, 400% and 8% for programs 3A–7A respectively, i.e. four overestimates out of five, but with higher %error values.

The M-L estimate is used as a reference when applying the experience-based multiplicative factor approach below. The results from the M-L estimates in this experiment differ from earlier investigations where the M-L method has provided consequent underestimates (Vander Wiel and Votta, 1993; Wohlin et al., 1995). It is assumed that this relates to the fact that in this experiment the reviewers cover close to 100% of the defects, see Table 9.

## 4.3 Filter with multiplicative factor

In the filtering approach, the estimations are performed in a sequence where the former program(s) are employed as experience base. In the analysis reported in Table 6, the three last estimates are used as experience base. In the text reading experiment (Wohlin et al., 1995), the experience-based factor is 2.11. This factor is used for the initial analysis and then an experience-based factor is derived for C-code inspections. The initial value is not included in the experience base since it comes from another type of inspection objects. The calculations behind Table 6 are explained for program 5A.

In the inspection of program 5A, 8 defects were found by a single reviewer ($n_{Cl\ 1}$) and 7 defects where found by more than one reviewer ($n_{Cl\ 2}$). The estimate of class 1 defects ($\hat{N}_{Cl\ 1}$) is then calculated as the number of found defects multiplied with the experience factor (8*1.2Ý10). The class 2 estimate ($\hat{N}_{Cl\ 2}$) is determined using the M-L estimator, which estimates the number of defects to be 7. The estimated total number of defects ($\hat{N}$) is 10+7=17 defects, while the correct number ($N$) is 16, i.e. the error is 1 defect overestimate. The percentage error is $100((17-16)/16) = 6\%$. The new experience factor is calculated based on data for class 1 from the three last inspections (3A, 4A and 5A). The contribution from program 3A to the factor is according to formula (2): (22–4)/14 = 18/14; from 4A: (16–8)/7= 8/7; and from 5A: (16–7)/8 = 9/8. The new factor is calculated as the average of those three.

**TABLE 6. Experience-based filter estimates**

| **Program** | ***n*** | | $\hat{N}$ | | ***N*** | $\hat{N} - N$ | **%error** | **New factor** |
|---|---|---|---|---|---|---|---|---|
| | **Cl 1** | **Cl 2** | **Cl 1** | **Cl 2** | | | | |
| 3A | 14 | 4 | 30 | 4 | 22 | 12 | +54% | $\frac{18}{14} = 1,29$ |
| 4A | 7 | 8 | 9 | 8 | 16 | 1 | +6% | $\frac{1}{2}\left(\frac{18}{14} + \frac{8}{7}\right) = 1,2$ |
| 5A | 8 | 7 | 10 | 7 | 16 | 1 | +6% | $\frac{1}{3}\left(\frac{18}{14} + \frac{8}{7} + \frac{9}{8}\right) = 1,18$ |
| 6A | 19 | 14 | 23 | 14 | 35 | 2 | +6% | $\frac{1}{3}\left(\frac{8}{7} + \frac{9}{8} + \frac{21}{19}\right) = 1,16$ |
| 7A | 15 | 5 | 17 | 5 | 20 | 2 | +10% | $\frac{1}{3}\left(\frac{9}{8} + \frac{21}{19} + \frac{15}{15}\right) = 1,11$ |
| | | | | | | Mean[a] | +7.0% | |
| | | | | | | St.dev | 2.0% | |

a. The mean and the standard deviation are calculated over programs 4A-7A since 3A is based on experience from another type of inspection object.

The analysis of the filter approach results in rather stable estimates, one or two defects over correct value. The mean error over the four experience-based estimates (4A-7A) is 7.0% and the standard deviation 2.0%. The reason for excluding the first value is that this estimate is based on experience from another type of inspection object. For the same reason, the factor used initially in the 3A estimation is not included in the experience base.

For the analysis without syntax defects, the percentage errors are 42%, 14%, 25%, 6% and 8% for programs 3A–7A respectively.

Figure 2 shows a comparison of the absolute percentage errors between the pure M-L and the filter estimates. The filter estimate of program 3A is excluded from the mean and standard deviation since it is based on experience from other circumstances. It can be concluded that the experience-based estimates in programs 4A-7A deviate less from the correct value than the M-L estimates do. The mean values are 10.5% and 7.0% for the M-L and filter respectively and the standard deviations are 10.7% and 2.0% respectively.

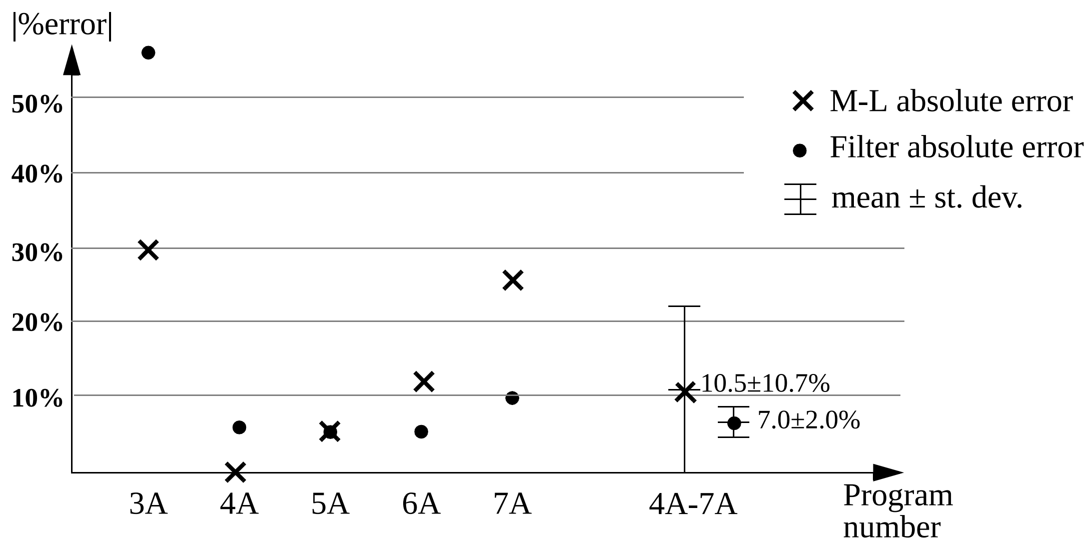

**FIGURE 2. Percentage error for M-L and filter estimates respectively.**

Even though the mean error is lower in the filter estimates than in the M-L, the difference between the methods is not statistically significant, see Table 7. Neither a Wilcoxon rank test nor a t-test for correlated samples give significant results (Siegel and Castellan, 1988; Montgomery, 1991). However according to an F-test, the variance is significantly lower in the filter method at a significance level of 0.05.

**TABLE 7. Statistical analysis M-L versus filter estimates**

| Study quantity | Test | Test parameter | Degr. of freedom | Significance level |
|---|---|---|---|---|
| Mean | Wilcoxon rank | $T^{-}$=2 | 3 | Not significant |
| Mean | t-test | t=0.787 | 3 | Not significant |
| Standard deviation | F-test | f= 28.42 | 3 | 0.05 |

## 4.4 Sensitivity analysis maximum-likelihood

In order to analyse how sensitive the estimates are to the composition of the inspection teams, a sensitivity analysis is conducted. If an estimate is very sensitive to the contribution of a single reviewer, the estimates will vary depending on the composition of the review team. This will probably disturb the defect content estimates. The ideal estimation technique will give the same estimate independently of changes in the inspection team.

The sensitivity analysis is performed by removing the reviewers, one by one, from the inspection team and performing an analysis on the remaining defect data. In Figure 3 the percentage error is plotted related to which reviewer is removed in the sensitivity analysis. To the left, the reviewer who found least defect is removed and to the right, the reviewer who found most defects is removed. The analysis data are presented in Table 8. For example, the data point (1,+36%) corresponds to the estimate of program 3A with reviewer B removed, which has the lowest rating (1) and the %error is +36%.

FIGURE 3. Percentage error related to the rating of reviewer removed in the sensitivity analysis of M-L estimates.

TABLE 8. Sensitivity analysis Maximum-Likelihood.

| Program | $n$ | $\hat{N}$ | $N$ | $\hat{N} - N$ | %error | Reviewer removed | |
|---|---|---|---|---|---|---|---|
| | | | | | | ID | Rating |
| 3A | 12 | 17 | | -5 | -23% | D | 5 |
| | 15 | 27 | | 5 | +23% | H | 4 |
| | 15 | 21 | 22 | -1 | -5% | F | 3 |
| | 17 | 34 | | 12 | +55% | C | 2 |
| | 17 | 30 | | 8 | +36% | B | 1 |
| 4A | 12 | 12 | | -4 | -25% | D | 5 |
| | 14 | 18 | | 2 | +12,5% | G | 4 |
| | 13 | 14 | 16 | -2 | -12,5% | E | 3 |
| | 15 | 17 | | 1 | +6% | H | 2 |
| | 14 | 15 | | -1 | -6% | F | 1 |
| 5A | 12 | 14 | | -2 | -12,5% | A | 5 |
| | 13 | 15 | | -1 | -6% | B | 4 |
| | 13 | 15 | 16 | -1 | -6% | H | 3 |
| | 14 | 17 | | 2 | +6% | F | 2 |
| | 15 | 18 | | 3 | +12,5% | C | 1 |
| 6A | 23 | 27 | | -8 | -20% | A | 5 |
| | 33 | 49 | | 16 | +40% | D | 4 |
| | 30 | 38 | 35 | 3 | +9% | E | 3 |
| | 30 | 37 | | 2 | +6% | G | 2 |
| | 31 | 35 | | 0 | 0% | C | 1 |
| 7A | 13 | 15 | | -5 | -25% | A | 5 |
| | 17 | 22 | 20 | 2 | +10% | G | 4 |
| | 18 | 25 | | 5 | +25% | B | 2 |
| | 17 | 20 | | 0 | 0% | E | 1 |

The sensitivity analysis of the M-L estimate reveals that the estimate is rather sensitive to the composition of inspection teams. Removing the reviewer with highest detection yield gives results of up to 25% underestimate. Removing the one with lowest detection yield gives estimates between correct and 36% overestimation. Program 3A can be studied in Table 8 as an extreme example of the sensitivity of the M-L method to the composition of inspection teams.

The average of the absolute percentage error of estimates for programs 4A-7A with one reviewer removed is 12.6%, with 10.2% standard deviation. The variation over different programs is shown in Figure 4.

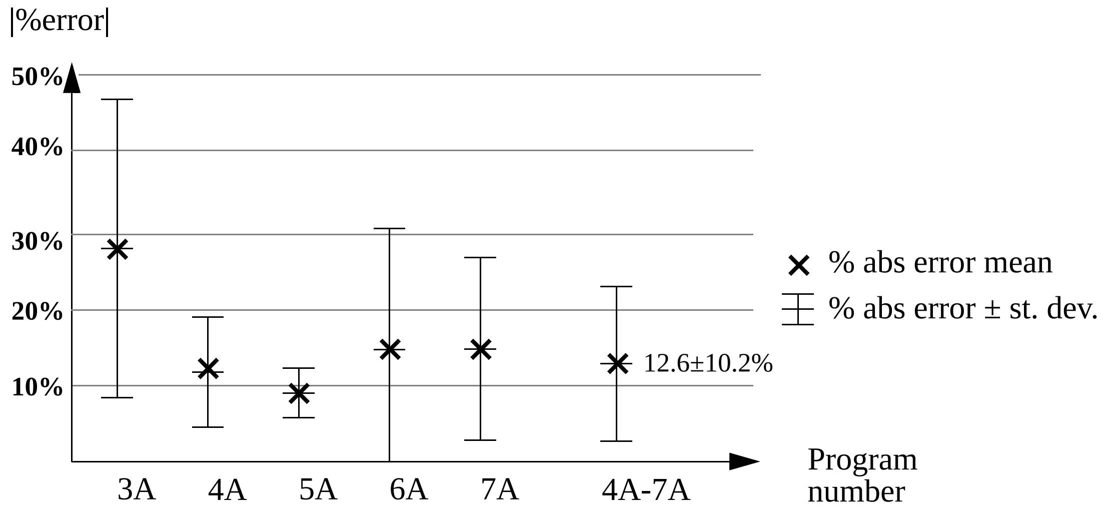

**FIGURE 4. Sensitivity analysis of M-L estimates**

One reason behind the variation in the sensitivity analysis is that one of the preconditions for the M-L estimate is not fulfilled: it is assumed that the probability for each reviewer to find the defects is the same. The detection yield in the experiment is calculated for all combinations of reviewers and programs as the number of defects found by the reviewer in the program, divided by the total number of defects in that program, see Table 9.

**TABLE 9. Defect detection yields for reviewers A-H**

| | A | B | C | D | E | F | G | H | All |
|---|---|---|---|---|---|---|---|---|---|
| 3A | - | 0.136 | 0.182 | 0.364 | - | 0.182 | - | 0.273 | 0.818 |
| 4A | - | - | - | 0.625 | 0.313 | 0.125 | 0.437 | 0.188 | 0.937 |
| 5A | 0.437 | 0.375 | 0.188 | - | - | 0.313 | - | 0.375 | 0.937 |
| 6A | 0.571 | - | 0.057 | 0.343 | 0.343 | - | 0.315 | - | 0.943 |
| 7A | 0.550 | 0.350 | - | - | 0.300 | - | 0.400 | - | 1.000 |
| Total | 0.535 | 0.276 | 0.123 | 0.411 | 0.323 | 0.204 | 0.366 | 0.278 | |

The yield varies between 0.057 and 0.57 for single programs, i.e. a factor 10, and between 0.123 and 0.535 for the total yield for reviewers, a factor 4.3. The experience-based filtering approach is introduced to overcome these variations.

## 4.5 Sensitivity analysis filter with multiplicative factor’

The sensitivity analysis for the filtering approach with the experience based factor is conducted as for the analysis of the approach with M-L estimates only: the reviewers are removed, one by one, and an estimation is performed with the limited set of reviewers. In Figure 5 the percentage error is plotted related to which reviewer is removed in the sensitivity analysis. To the left, the least contributing reviewer is removed and to the right, the most contributing reviewer is removed. In Table 10, the complete analysis data are presented.

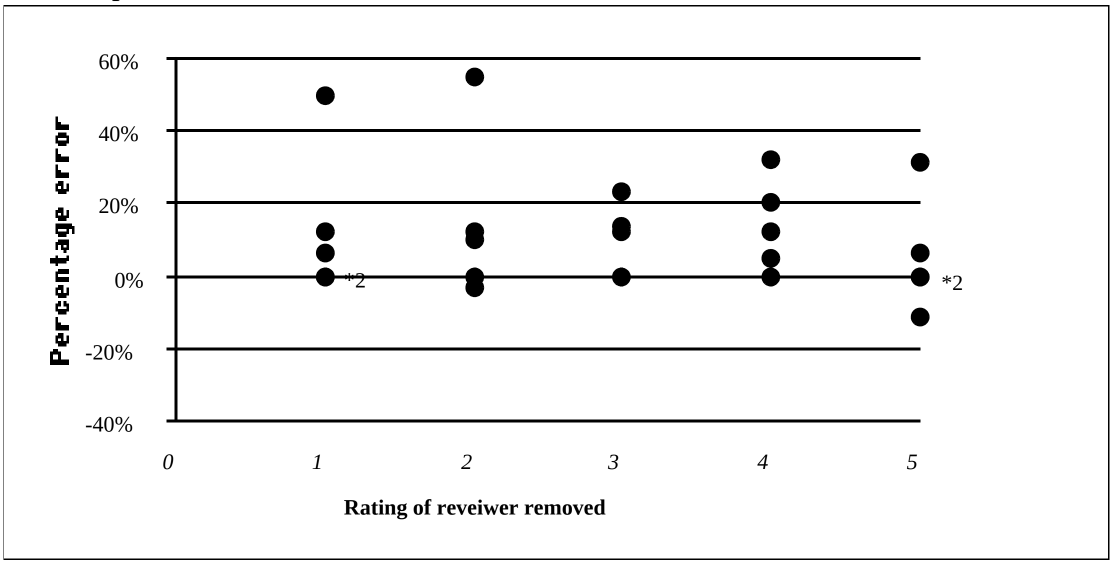

**FIGURE 5. Percentage error related to the rating of reviewer removed in the sensitivity analysis of filter estimates.**

In the analysis, the multiplicative factor for program 3A is set to 2.11 and thereafter it is based on the previous analyses for the same type of inspection team. For example, the factor for analysis of program 4A, the reviewer with the highest rating removed, is based on the outcome of the analysis of program 3A with the reviewer with the highest rating removed.

TABLE 10. Sensitivity analysis, filter.

| Prog | $n$ |  | $\hat{N}$ |  | $N$ | $\hat{N} - N$ | %error | New factor | Reviewer removed |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  | Cl 1 | Cl 2 | Cl 1 | Cl 2 |  |  |  |  | ID | Rating |
| 3A | 9 | 3 | 19 | 3 |  | 0 | 0% | 2.11 | D | 5 |
|  | 13 | 2 | 27 | 2 |  | +7 | +32% | 1.54 | H | 4 |
|  | 11 | 4 | 23 | 4 | 22 | +5 | +23% | 1.64 | F | 3 |
|  | 15 | 2 | 32 | 2 |  | +12 | +55% | 1.33 | C | 2 |
|  | 14 | 3 | 30 | 3 |  | +11 | +50% | 1.36 | B | 1 |
| 4A | 8 | 4 | 17 | 4 |  | +5 | +31% | 1.8 | D | 5 |
|  | 8 | 6 | 12 | 6 |  | +2 | +12% | 1.39 | G | 4 |
|  | 5 | 8 | 8 | 8 | 16 | 0 | 0% | 1.62 | E | 3 |
|  | 8 | 7 | 11 | 7 |  | +2 | +12% | 1.23 | H | 2 |
|  | 7 | 7 | 10 | 7 |  | +1 | +6% | 1.32 | F | 1 |
| 5A | 6 | 6 | 11 | 6 |  | +1 | +6% | 1.76 | A | 5 |
|  | 7 | 6 | 10 | 6 |  | 0 | 0% | 1.41 | B | 4 |
|  | 8 | 5 | 13 | 5 | 16 | +2 | +12% | 1.54 | H | 3 |
|  | 8 | 6 | 10 | 6 |  | 0 | 0% | 1.24 | F | 2 |
|  | 9 | 6 | 12 | 6 |  | +2 | +12% | 1.25 | C | 1 |
| 6A | 11 | 12 | 19 | 12 |  | -4 | -11% | 1.87 | A | 5 |
|  | 22 | 11 | 31 | 11 |  | +7 | +20% | 1.30 | D | 4 |
|  | 19 | 11 | 29 | 11 | 35 | +5 | +14% | 1.44 | E | 3 |
|  | 18 | 12 | 22 | 12 |  | -1 | -3% | 1.25 | G | 2 |
|  | 16 | 15 | 20 | 15 |  | 0 | 0% | 1.25 | C | 1 |
| 7A | 8 | 5 | 15 | 5 |  | 0 | 0% | 1.87 | A | 5 |
|  | 13 | 4 | 17 | 4 | 20 | +1 | +5% | 1.28 | G | 4 |
|  | 14 | 4 | 18 | 4 |  | +2 | +10% | 1.34 | B | 2 |
|  | 12 | 5 | 15 | 5 |  | 0 | 0% | 1.25 | E | 1 |

The variation in the filter estimates over different programs, compared to the M-L estimates is shown in Figure 6. The figure shows the mean values and standard deviations for the different analyses with one reviewer removed. The staples to the right shows the mean and standard deviation for the estimates with one reviewer removed for all the programs 4A-7A.

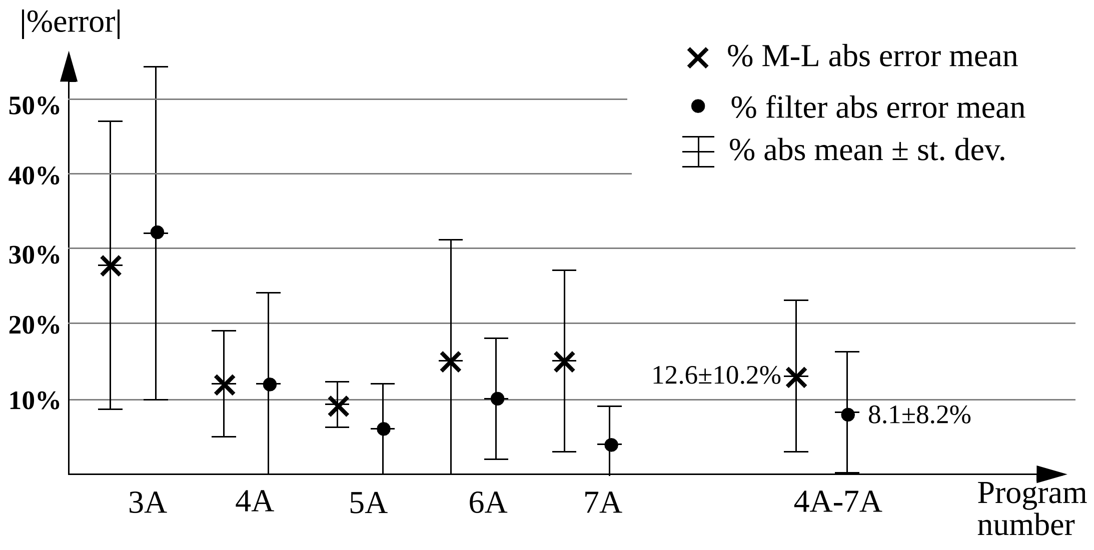

**FIGURE 6. Sensitivity analysis of experience-based filter estimates compared to M-L estimates.**

The sensitivity analysis shows that the experience-based estimates are less sensitive to variations than the M-L estimates. The absolute mean error compared to correct value is 12.6% and 8.1% for the M-L and filter estimates respectively, and the standard deviation is 10.2% and 8.2% respectively. The differences in favour of the filter method are statistically significant, see Table 11. Wilcoxon rank test and the t-test judges the difference in the mean value on a significance level close to 0.1. The F-test judges the variance to be smaller with the filter method on a significance level of 0.05. This means that the $H_0$ hypothesis can be rejected with respect to less sensitivity to inspection team composition.

**TABLE 11. Statistical analysis M-L versus filter estimates**

| Study quantity | Test | Test parameter | Degr. of freedom | Significance level |
|---|---|---|---|---|
| Mean | Wilcoxon rank | $T^{-}$=48 | 17 | 0.09 |
| Mean | t-test | t=1.54 | 18 | 0.1 |
| Standard deviation | F-test | f= 3.0 | 18 | 0.05 |

Another important issue to notice from the experiment is that the multiplicative factor depends on the reviewer profile. The analysis based on all but the highest rate reviewer (5) has a factor in the range of 1.7-2.1, see Table 10 and Figure 7, while the analysis based on all but lowest rate reviewer (1) has a factor in the range of 1.2-1.4. Hence the factor differs with the contribution of the group members.

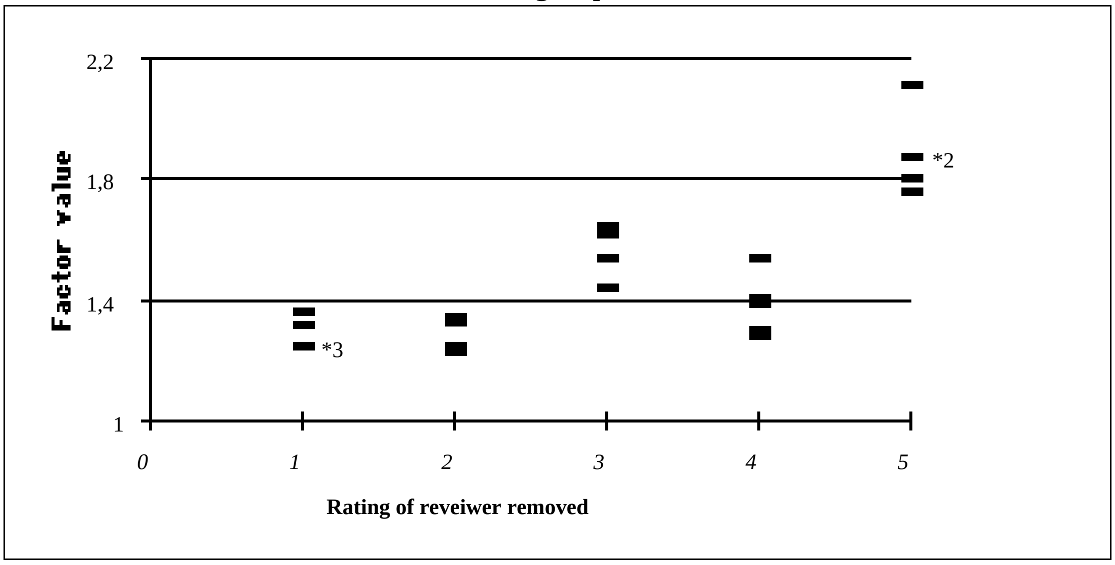

**FIGURE 7. Value of multiplicative value versus rating of reviewer removed.**

## 5. Conclusion and further work

It can be concluded that capture-recapture estimates using M-L suffer from the same problems in this experiment as reported before (Vander Wiel and Votta, 1993; Wohlin et al., 1995), i.e. the precondition of the same detection probabilities is not fulfilled, hence the estimates fail. The difference in this study is that the earlier reported underestimates are changed into overestimates. Furthermore the sensitivity analysis shows that the method is sensitive to changing the composition of inspection teams.

The $H_0$ hypothesis can not be rejected as it is formulated in Section 3.1. However, two out of three sub-hypotheses can be rejected: 1) The mean estimate is *not* significantly better with the filter method. 2) The variance *is* significantly less in the filter analysis at a level of 0.05. 3) Applying the filter technique and multiplicative factor gives estimates which *are less sensitive* to changes in the review team, i.e. the mean percentage error and the variance are significantly less at a significance level of 0.1 and 0.05 respectively.

The filtering technique provides a very pragmatic application method: Multiply the number of defects found in class 1 with a factor and apply the M-L estimate to the defects found in class 2.

The key issue in this method is the multiplicative factor. The factor is dependent on the composition of the inspection team. It can be concluded from the sensitivity analysis that the factor is higher for the team with the highest detection yield reviewer removed, than for the team with the lowest detection yield reviewer removed. The factor is 1.7-2.1 in the first case and 1.2-1.4 in the latter case. The experience-based filter approach, which is proposed in this paper, takes this into account as long as the reviewer profile is stable over different inspection occasions.

Future enhancement of the estimation method will be in the area of correlating the multiplicative value to some sort of characterization of the reviewers. A possible solution is to define three types of reviewer teams, low, medium, and high, based on characterization of their experience, performance in earlier inspections, time spent in the current inspection etc. To each of the team types, an experience based multiplicative factor is related which is applied when estimating defects after an inspection by that type of team.

In this experiment it is concluded that the multiplicative factor in the sensitivity analysis changed depending on whether the reviewer with high or low detection yield is removed. Thus the characterization of the group of reviewers would change and another multiplicative value should be selected.

Another approach to reducing the variation of individuals' contribution is to remove the reviewers with highest and lowest detection yields and conduct the estimate based on the reviewers in the middle. The drawback is that we lose data, but the advantage is that the outliers in terms of detection yield are removed.

Further work with this approach is to find a way to characterize the reviewer group and to empirically evaluate the outcome. A first step might be to debrief the reviewers in the study to find factors affecting their performance.

As in all empirical studies, the need for replication of results can not be stressed enough. The PSP environment utilized in this experiment enhances the possibility of replicating the work.

## Acknowledgement

We would like to thank the anonymous Journal of Empirical Software Engineering reviewers which have substantially improved the paper. We also want to express our gratitude to our colleagues at Lund University, Department of Communication Systems and Q-Labs, Lund, for taking part in the experiment and giving valuable comments to the report. This work was partly funded by The Swedish National Board for Industrial and Technical Development (NUTEK), grant 1K1P-97-09690 and 1K1P-97-09673.

## References

Briand, L. C., El Emam, K., Freimut, B., and Laitenberger, O. 1997. Quantitative Evaluation of Capture-Recapture Models to Control Software Inspections. *Proceedings 8th International Symposium on Software Reliability Engineering*, Albuquerque, New Mexico, USA, 234–244.

Briand, L. C., El Emam, K., and Freimut, B. 1998. Comparison and Integration of Capture-Recapture Models and the Detection Profile Method. To appear in *Proceedings 9th International Symposium on Software Reliability Engineering*, Paderborn, Germany.

Chillarege, R., Bhandari, I., Chaar, J. Halliday, M., Moebus, D., Ray, B., and Wong, Man-Yuen. 1992. Orthogonal Defect Classification-A Concept for In-Process Measurements. *IEEE Transactions on Software Engineering*, 18(11): 943-956.

Ebrahimi, N. B.1997. On the Statistical Analysis of the Number of Errors Remaining in a Software Design Document after Inspection. *IEEE Transactions on Software Engineering*, 23(8):529–532.

Eick, S., Loader, C., Long, D., Votta, L., and Vander Wiel, S. 1992. Estimating Software Fault Content Before Coding. *Proceedings 14th International Conference on Software Engineering*, Melbourne, Australia, 59-65.

Fagan, M. 1986. Advances in Software Inspections. *IEEE Transactions on Software Engineering*, 12(7): 744-751.

Gilb, T., and Graham, D. 1993. *Software Inspection*. Addison-Wesley.

Humphrey, W. 1995. *A Discipline for Software Engineering.* Addison-Wesley.

Johnson, P. M., and Tjahjono, D. 1998. Does Every Inspection Really Need a Meeting. *Journal of Empirical Software Engineering,* 3(1): 9–35.

Otis, D., Burnham G., White, G., and Anderson, D. 1978. Statistical Inference from Capture Data on Closed Animal Populations. *Wildlife Monografics*, 62: 1–135.

Siegel, S., and Castellan, J. Jr. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill.

Vander Wiel, S., and Votta, L. 1993. Assessing Software Designs using Capture-Recapture Methods. *IEEE Transactions on Software Engineering*, 19(11): 1045-1054.

Votta, L. 1993. Does every inspection need a meeting? *Proceedings of ACM SIGSOFT ‘93 Symposium on Foundations of Software Engineering,* New Orleans, LA, USA, 107-114.

Porter, A., Siy, H., Toman, C. A., and Votta, L.1995. An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development. *Proceedings of ACM SIGSOFT ‘95 Symposium on Foundations of Software Engineering*, Washington, DC, USA, 92-103.

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

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

Wohlin, C., and Runeson, P. 1998. Defect Content Estimations from Review Data. *Proceedings 20th International Conference on Software Engineering*, Japan, 400-409.

# Annex: Defect detection data

**TABLE 12. Defect data program 3A**

| Defect no | Type | B | C | D | F | H |
|---|---|---|---|---|---|---|
| 1 | 20 | 0 | 0 | 0 | 0 | 0 |
| 2 | 20 | 0 | 0 | 1 | 0 | 0 |
| 3 | 20 | 0 | 0 | 1 | 0 | 0 |
| 4 | 20 | 0 | 0 | 1 | 0 | 0 |
| 5 | 20 | 0 | 0 | 1 | 0 | 0 |
| 6 | 40 | 1 | 1 | 1 | 1 | 1 |
| 7 | 80 | 1 | 0 | 0 | 0 | 0 |
| 8 | 20 | 0 | 0 | 1 | 0 | 0 |
| 9 | 80 | 1 | 1 | 0 | 0 | 0 |
| 10 | 80 | 0 | 0 | 0 | 0 | 0 |
| 11 | 80 | 0 | 0 | 0 | 0 | 0 |
| 12 | 20 | 0 | 0 | 0 | 0 | 0 |
| 13 | 80 | 0 | 0 | 0 | 1 | 0 |
| 14 | 20 | 0 | 0 | 0 | 1 | 0 |
| 15 | 80 | 0 | 0 | 0 | 1 | 0 |
| 16 | 100 | 0 | 0 | 1 | 0 | 1 |
| 17 | 10 | 0 | 0 | 1 | 0 | 0 |
| 18 | 100 | 0 | 0 | 0 | 0 | 1 |
| 19 | 20 | 0 | 1 | 0 | 0 | 1 |
| 20 | 20 | 0 | 0 | 0 | 0 | 1 |
| 21 | 20 | 0 | 0 | 0 | 0 | 1 |
| 22 | 10 | 0 | 1 | 0 | 0 | 0 |
| **Sum** | | **3** | **4** | **8** | **4** | **6** |

**TABLE 13. Defect data program 4A**

| Defect no | Type | D | E | F | G | H |
|---|---|---|---|---|---|---|
| 1 | 20 | 1 | 0 | 0 | 0 | 0 |
| 2 | 20 | 1 | 0 | 0 | 0 | 0 |
| 3 | 40 | 1 | 0 | 0 | 1 | 0 |
| 4 | 20 | 1 | 1 | 0 | 1 | 0 |
| 5 | 20 | 1 | 1 | 0 | 1 | 0 |
| 6 | 20 | 0 | 1 | 0 | 1 | 1 |
| 7 | 40 | 1 | 0 | 0 | 0 | 1 |
| 8 | 20 | 0 | 0 | 0 | 0 | 0 |
| 9 | 80 | 0 | 0 | 1 | 0 | 0 |
| 10 | 80 | 1 | 0 | 0 | 1 | 0 |
| 11 | 20 | 1 | 0 | 0 | 1 | 1 |
| 12 | 90 | 1 | 0 | 0 | 0 | 0 |
| 13 | 20 | 1 | 0 | 1 | 0 | 0 |
| 14 | 20 | 0 | 0 | 0 | 1 | 0 |
| 15 | 80 | 0 | 1 | 0 | 0 | 0 |
| 16 | 80 | 0 | 1 | 0 | 0 | 0 |
| **Sum** | | **10** | **5** | **2** | **7** | **3** |

**TABLE 14. Defect data program 5A**

| Defect no | Type | A | B | C | F | H |
|---|---|---|---|---|---|---|
| 1 | 20 | 0 | 0 | 1 | 0 | 1 |
| 2 | 20 | 0 | 0 | 0 | 0 | 0 |
| 3 | 20 | 1 | 0 | 0 | 0 | 0 |
| 4 | 20 | 1 | 0 | 0 | 1 | 1 |
| 5 | 20 | 0 | 1 | 1 | 1 | 0 |
| 6 | 20 | 1 | 0 | 0 | 0 | 0 |
| 7 | 20 | 1 | 1 | 1 | 1 | 0 |
| 8 | 20 | 1 | 1 | 0 | 0 | 1 |
| 9 | 20 | 0 | 1 | 0 | 1 | 0 |
| 10 | 20 | 0 | 1 | 0 | 0 | 0 |
| 11 | 80 | 0 | 1 | 0 | 0 | 0 |
| 12 | 100 | 1 | 0 | 0 | 0 | 1 |
| 13 | 20 | 1 | 0 | 0 | 0 | 0 |
| 14 | 10 | 0 | 0 | 0 | 0 | 1 |
| 15 | 50 | 0 | 0 | 0 | 1 | 0 |
| 16 | 20 | 0 | 0 | 0 | 0 | 1 |
| **Sum** | | **7** | **6** | **3** | **5** | **6** |

**TABLE 15. Defect data program 6A**

| Defect no | Type | A | C | D | E | G |
|---|---|---|---|---|---|---|
| 1 | 20 | 0 | 0 | 0 | 0 | 1 |
| 2 | 20 | 0 | 0 | 1 | 1 | 1 |
| 3 | 20 | 1 | 0 | 1 | 1 | 1 |
| 4 | 20 | 0 | 0 | 0 | 0 | 0 |
| 5 | 20 | 1 | 0 | 1 | 0 | 1 |
| 6 | 20 | 1 | 0 | 0 | 1 | 0 |
| 7 | 20 | 1 | 0 | 0 | 1 | 0 |
| 8 | 100 | 0 | 0 | 0 | 0 | 1 |
| 9 | 40 | 0 | 0 | 0 | 0 | 0 |
| 10 | 20 | 0 | 0 | 1 | 1 | 0 |
| 11 | 20 | 1 | 0 | 1 | 1 | 0 |
| 12 | 20 | 1 | 0 | 1 | 1 | 0 |
| 13 | 20 | 0 | 0 | 1 | 1 | 0 |
| 14 | 20 | 1 | 0 | 1 | 1 | 0 |
| 15 | 20 | 0 | 0 | 1 | 0 | 1 |
| 16 | 20 | 0 | 0 | 1 | 0 | 1 |
| 17 | 100 | 1 | 0 | 0 | 0 | 0 |
| 18 | 100 | 1 | 0 | 0 | 0 | 0 |
| 19 | 100 | 1 | 0 | 0 | 0 | 0 |
| 20 | 100 | 1 | 0 | 0 | 0 | 0 |
| 21 | 20 | 1 | 0 | 0 | 0 | 0 |
| 22 | 80 | 1 | 0 | 0 | 0 | 0 |
| 23 | 80 | 1 | 0 | 0 | 0 | 0 |
| 24 | 80 | 1 | 0 | 0 | 0 | 0 |
| 25 | 80 | 1 | 0 | 0 | 0 | 0 |
| 26 | 80 | 1 | 0 | 0 | 0 | 0 |
| 27 | 20 | 1 | 0 | 1 | 0 | 1 |
| 28 | 80 | 1 | 0 | 0 | 0 | 1 |
| 29 | 20 | 1 | 0 | 1 | 0 | 1 |
| 30 | 50 | 0 | 0 | 0 | 0 | 1 |
| 31 | 80 | 0 | 0 | 0 | 1 | 0 |
| 32 | 80 | 0 | 0 | 0 | 1 | 0 |
| 33 | 80 | 0 | 0 | 0 | 1 | 0 |
| 34 | 80 | 0 | 1 | 0 | 0 | 0 |
| 35 | 80 | 0 | 1 | 0 | 0 | 0 |
| **Sum** | | **20** | **2** | **12** | **12** | **11** |

**TABLE 16. Defect data program 7A**

| Defect no | Type | A | B | E | G |
|---|---|---|---|---|---|
| 1 | 80 | 1 | 1 | 0 | 1 |
| 2 | 80 | 1 | 1 | 1 | 1 |
| 3 | 80 | 1 | 1 | 1 | 1 |
| 4 | 80 | 1 | 1 | 1 | 1 |
| 5 | 20 | 0 | 1 | 0 | 0 |
| 6 | 80 | 0 | 1 | 0 | 0 |
| 7 | 80 | 0 | 1 | 0 | 1 |
| 8 | 20 | 1 | 0 | 0 | 0 |
| 9 | 80 | 1 | 0 | 0 | 0 |
| 10 | 20 | 1 | 0 | 0 | 0 |
| 11 | 20 | 1 | 0 | 0 | 0 |
| 12 | 20 | 1 | 0 | 0 | 0 |
| 13 | 80 | 1 | 0 | 0 | 0 |
| 14 | 20 | 1 | 0 | 0 | 0 |
| 15 | 50 | 0 | 0 | 0 | 1 |
| 16 | 80 | 0 | 0 | 0 | 1 |
| 17 | 20 | 0 | 0 | 0 | 1 |
| 18 | 80 | 0 | 0 | 1 | 0 |
| 19 | 80 | 0 | 0 | 1 | 0 |
| 20 | 80 | 0 | 0 | 1 | 0 |
| **Sum** | | **11** | **7** | **6** | **8** |