Software Testing MTAT.03.159
Lab 4: Software Inspection and Fault Estimation

Dietmar Pfahl, University of Tartu, Inst. of Computer Science,
based on materials developed by
Carina Andersson, Yeni Li Helgesson, Per Runeson and Thomas Thelin,
Dept. Computer Science, Lund University, Sweden

May 7, 2013

1 Introduction

Software inspection is an important static techniques to detect faults early in the software life-cycle. The purpose of inspections is to manually scrutinize a software artefact, for example, requirements, design or code. In addition to inspections, estimation techniques can be applied in order to estimate the fault content. A software inspection process with estimation points is shown in Figure 1. You will perform preparation, subjective estimation, compilation, objective estimation, meeting, and discussion of subjective and objective estimation in this exercise.

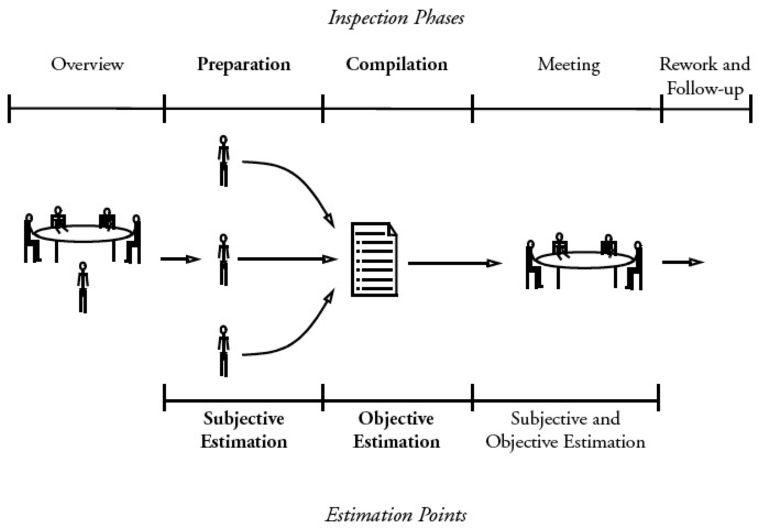

Figure 1: The phases of the inspection process in combination with estimation points.

2 Learning Objectives

The exercise aims at giving an understanding of software inspections and estimation techniques. The specific learning goal is to try an inspection technique and to apply fault content estimators to the inspection process.

3 Tasks to Perform

Download the documents of the *taxi system* posted on the course wiki.

Task 1 - individually: Read the *guideline and use cases*. Inspect the *requirements specification (RSpec)* using the method *usage-based reading*. It is important that you do the inspection of the RSpec *individually*! Log the time you spend on inspecting and log the faults you find in the RSpec. In the log of the faults, include a brief description of the kind of fault and the fault's location in the requirements document. When you are finished, based on your intuition and your experience from inspecting the document, estimate the number of remaining faults (subjectively). Specify the minimum, maximum and most probable number of faults remaining in RSpec. (Note: the true number of faults is not known).

Task 2: Before you start this task, form groups of two students (pairs). If one student is left over in the lab group, form one group of 3 students. Compare your individual fault logs and compile a joint fault log identifying the common faults found, the faults that only one of you found and false positives (i.e., faults that one of you considered to be a fault but after discussion turns out to be not a fault). In the group of three students also identify those faults that were found by exactly two students in the group.

Task 3: Revisit your individual subjective estimates of the total number of faults (or the number of remaining - undetected - faults) and make a new (subjective) estimate of the number of remaining faults. Explain how you established the group estimate.

Task 4: Use the Mt capture-recapture estimation model (see lecture slides) to estimate the number of remaining faults based on the data that you compiled in Task 2. Discuss whether the Mt model is appropriate (have a look at the underlying assumptions of the Mt model - see Appendix).

Task 5: Now assume that there is an additional reviewer who's fault data is merged with the fault log you produced in Task 2. The fault log produced in Task 2 contains the following subsets of faults: F1 is the set of faults detected by exactly on reviewer (but not by both), F2 is the set of faults detected by exactly two reviewers, and if you are the group of 3 students, F3 is the set of faults detected by all three students. If you are a group of 2 students assume the following: the (virtual) third reviewer found n1=rounded((number of faults in F1)/3) new faults, n2=rounded((number of faults in F2)/3) faults that are duplicates of faults in F1, and n3=rounded((number of faults in F2)/2) faults that are duplicates of faults in F2. If you are the group of 3 students, assume the following: the (virtual) fourth reviewer found n1=rounded((number of faults in F1)/4) new faults, n2=rounded((number of faults in F2)/4) faults that are duplicates of faults in F1, n3=rounded((number of faults in F2)/4) faults that are duplicates of faults in F2, and n4=rounded((number of faults in F3)/3) are duplicates of faults in F3.

Task 6: Discuss whether there are other ways than capture-recapture models to estimate the fault content when doing inspections (reviews). What metrics should be collected in order to trust these other types estimates?

Bonus Task: Download the Scilab software tool (an open software comparable to Matlab) from the course web or from https://www.scilab.org/. Use the Scilab files for the capture-recapture models M0, Mt, and Mh and repeat Tasks 4 and 5 but produce estimates with all three estimation models (in each task). Discuss which model is most appropriate. Read the Appendix for details on Scilab and the capture-recapture model functions.

4 Report

The purpose of the report is to summarise and dicuss the results of this lab. Submission deadline is before the begin of next week's lab session (i.e., individual deadline for each lab goup). Please submit one report per student pair (or triple). Only PDF files submitted via the submit function provided on the course wiki will be accepted. Late submission up to 24 hours after deadline will receive a penalty of 4 marks (50%). Submissions that are more than 24 hours late will not be looked at and automatically receive 0 marks. The total number of marks for the lab report is 8 marks. The marks are distributed as follows: Tasks 1 and 2 together equal 4 marks, and Tasks 3 to 6 together equal 4 marks. The Bonus Task will give you up to 4 extra marks.

5 Appendix

Terminology

Variable	Explanation
D	Number of unique faults found
k	Number of reviewers
D_k	Number of unique faults found when k reviewers inspect
f_1	Number of faults found by one reviewer
f_x	Number of faults found by x reviewers

Assumptions for capture-recapture estimations

The restrictions for capture-recapture estimators (closed population) are:

1. Once the document is issued for inspection, it must not be changed; and the performance of the reviewers should be constant, i.e. given the same document the reviewers should find the same faults.
2. Reviewers must not reveal their proposed faults to other reviewers.
3. Reviewers must ensure that they accurately record and document every fault they find. Additionally, the inspection process, for example, at the collection meeting, must not discard any correct faults.
4. All reviewers must be provided with identical information, in terms of source materials, standards, inspection aids, etc.; and this material must be available to them at all times.

Assumption 4 also implies equality between reviewer abilities and the complexity of finding different faults. Depending on the degree of freedom of the ability of the reviewers and probability of the faults to be detected, four basic models are formed:

- M0, all faults have equal detection probability, all reviewers have equal detection ability.
- Mt, all faults have equal detection probability, reviewers may have different detection abilities.
- Mh, faults may have different detection probabilities, all reviewers have equal detection ability.
- Mth, faults may have different detection probabilities, reviewers may have different detection abilities.

Connected to each model, there are a number of estimators, see Table 1.

Models in capture-recapture

Mh-JK estimator (model Mh, estimator Jack-knife)

The Jackknife estimator is not based on a specific distribution. In Figure 2, the first five orders of Mh-JK is presented.

Table 1: Some models and estimators in capture-recapture for software inspections. The bold ones will be used in the exercise.

Model	**Estimators**
M0	**M0-ML – Maximum likelihood**
Mt	**Mt-ML – Maximum likelihood**
	Mt-Ch – Chao's estimator
Mh	**Mh-JK – Jackknife**
	Mh-Ch – Chao's estimator
Mth	Mth-Ch – Chao's estimator

Order	**Formula**
1	$\hat{N}_{J1} = D + \left(\frac{k-1}{k}\right) \cdot f_1$
2	$\hat{N}_{J2} = D + \left(\frac{2k-3}{k}\right) \cdot f_1 - \left\{\frac{(k-2)^2}{k(k-1)}\right\} \cdot f_2$
3	$\hat{N}_{J3} = D + \left(\frac{3k-6}{k}\right) \cdot f_1 - \left\{\frac{3k^2-15k+19}{k(k-1)}\right\} \cdot f_2 + \left\{\frac{(k-3)^3}{k(k-1)(k-2)}\right\} \cdot f_3$
4	$\hat{N}_{J4} = D + \left(\frac{4k-10}{k}\right) \cdot f_1 - \left\{\frac{6k^2-36k+55}{k(k-1)}\right\} \cdot f_2 + \left\{\frac{4k^3-42k^2+148k-175}{k(k-1)(k-2)}\right\} \cdot f_3 - \left\{\frac{(k-4)^4}{k(k-1)(k-2)(k-3)}\right\} \cdot f_4$
5	$\hat{N}_{J5} = D + \left(\frac{5k-15}{k}\right) \cdot f_1 - \left\{\frac{10k^2-70k+125}{k(k-1)}\right\} \cdot f_2 + \left\{\frac{10k^3-120k^2+485k+660}{k(k-1)(k-2)}\right\} \cdot f_3 - \left\{\frac{(k-4)^5-(k-5)^5}{k(k-1)(k-2)(k-3)}\right\} \cdot f_4 + \left\{\frac{(k-5)^5}{k(k-1)(k-2)(k-3)(k-4)}\right\} \cdot f_5$

Figure 2: The formulae of the five first subestimators (orders) of Mh-JK.

Confidence interval

A 95% nominal confidence interval means that 95% of the correct values should be within the interval in the ideal case. The 95% confidence interval, assuming normal distributions is calculated as:

$$\hat{N} - 1{,}96\sqrt{\mathrm{Var}(\hat{N})},\ \hat{N} + 1{,}96\sqrt{\mathrm{Var}(\hat{N})}$$

where $\hat{N}$ is the estimated number of faults. The 95% confidence interval using the log-normal distribution is calculated as:

$$D + \frac{\hat{N} - D}{C}, D + C(\hat{N} - D)$$

$$= \exp\left(1{,}96 \sqrt{\log\left(1 + \frac{\mathrm{Var}(\hat{N})}{(\hat{N} - D)^2}\right)}\right)$$

The confidence interval calculations are only implemented in Mh-JK.

Scilab files

m0mle.sce – Estimates a point estimation of the fault content using the model M0

Input: Inspection data represented in a matrix of ones (found fault) and zeroes (not found fault). Row_i=fault no. i, and $column_j$=reviewer no. j. An example of a matrix is shown below (3 reviewers and 7 faults found):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Output: Estimated number of faults.

mtmle.sce – Estimates a point estimation of the fault content using the model Mt.

Input: The same input as momle.sce

Output: The same output as momle.sce

capjke.sce – Estimates a point estimation and confidence interval of the fault content using the model Mh. The confidence interval is based on selection procedure of what order to use.

Input: The same input as momle.sce

Output: An array consisting of [estimated number of faults, standard deviation, confidence interval based on the log-normal distribution, confidence interval based on the normal distribution].

In order to execute the model functions you first need to load these files into the scilab console. You also need to load the helper function comb.sce which is called from some of the model functions. details about loading functions and input matrices can be found in the Scilab documentation and help function.