Exercise

Software inspection and estimation
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, a 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 are shown in Figure 1. You will perform preparation, compilation, subjective estimation and objective estimation in this exercise.

![Inspection Phases diagram](image)

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. Individual Preparation (Inspection)

The inspection and subjective estimation parts of the exercise should be done individually before the exercise in order to get enough data for the objective estimation.

Assignment 1: Read chapter 10 in [Burnstein03] and the slides from the lecture Inspections.

Assignment 2: Download the taxi system on the homepage of the course. Read the guideline and use cases. Inspect the requirements document using the method usage-based reading. Log the time you spent on inspecting the artefact and log the faults you find in the requirements document. When you are finished, try to estimate the number of remaining faults (subjectively) by specifying the minimum, maximum and most probable number of faults left. Note that the true number of faults are not known.
4. Exercise (Fault content estimation)

Assignment 3: Read appendix.

Assignment 4: Divide in groups of two. Compile the faults you have found by identifying the common faults found, the faults that only one of you found and false positives.

Assignment 5: Estimate the number of remaining faults. Use both a subjective method and the capture-capture methods. Discuss the results.

Assignment 6: Divide in groups of four. Compile the faults you have found by identifying the common faults found, the faults that only one of you found and false positives.

Assignment 7: Estimate the number of remaining faults. Use both a subjective method and the capture-recapture methods. Discuss the results.

Assignment 8: Discuss whether there are other ways to estimate the fault content after inspections. What metrics should be collected in order to trust the results?

5. Report

The purpose of the report is to discuss the result of the exercise and related topics. Write the name of exercise, your names, group number and email addresses on the first page of the report. The size of the report should be 4-5 Letter size pages (including the first page).

Write a report (1-2 Letter size-pages), where you discuss the results of the inspection and the estimation. Discuss the assignments and:

• Which method(s) works best and why?
• What does accurately mean in this case, i.e. when can you trust the results?
• Do the estimators estimate accurately? Why or why not?
• What decisions during software development can be taken based on the results?
• If not capture-recapture estimators are used, what metrics should be collected during the inspections process in order to estimate the fault content?

In addition, discuss the following related topics in the report (1-2 Letter size pages):

6. Appendix

Terminology

<table>
<thead>
<tr>
<th>Variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Number of unique faults found.</td>
</tr>
<tr>
<td>k</td>
<td>Number of reviewers</td>
</tr>
<tr>
<td>D_k</td>
<td>Number of unique faults found when k reviewers inspect.</td>
</tr>
<tr>
<td>f_1</td>
<td>Number of faults found by one reviewer.</td>
</tr>
<tr>
<td>f_x</td>
<td>Number of faults found by x reviewers.</td>
</tr>
</tbody>
</table>

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 2.
Models in capture-recapture

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Estimators</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0</td>
<td>M0-ML – Maximum likelihood</td>
</tr>
<tr>
<td>Mt</td>
<td>Mt-ML – Maximum likelihood</td>
</tr>
<tr>
<td></td>
<td>Mt-Ch – Chao’s estimator</td>
</tr>
<tr>
<td>Mh</td>
<td>Mh-JK – Jackknife</td>
</tr>
<tr>
<td></td>
<td>Mh-Ch – Chao’s estimator</td>
</tr>
<tr>
<td>Mth</td>
<td>Mth-Ch – Chao’s estimator</td>
</tr>
</tbody>
</table>

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

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

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

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

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:

$\hat{N} \pm 1.96 \sqrt{\text{Var}(\hat{N})}$
where $\hat{N}$ is the estimated number of faults. The 95% confidence interval using the log-normal distribution is calculated as:

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

$$\exp \left( 1.96 \log \left( 1 + \frac{\text{Var}(\hat{N})}{(\hat{N} - D)^2} \right) \right)$$

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

Matlab files

m0mle.m – 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):

```
1 0 0
0 1 1
1 1 1
1 0 0
0 0 1
0 1 0
1 1 1
```

*Output:* Estimated number of faults left.

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

*Input:* The same input as momle.m

*Output:* The same output as momle.m

capjke.m – 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 as m0mle.m

*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].

mhjke2.m – Estimates a point estimation and confidence interval of the fault content using the model Mh.
Input: An array consisting of [inspection data, order of Mh-JK, probability used in the confidence interval calculation (default = 0.95)].

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].

7. References