Software Testing MTAT.03.159

Lab 4: Software Inspection and Fault Estimation

Institute of Computer Science, University of Tartu

Software inspection is an important static technique 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.

The following activities need to be done for completing this assignment:

Part A (ca. 40-60 min): Individual work – don't interact with another person in the room!

1. Read and understand the requirements and the related specification document (see below).
2. Review the specification document against the requirements (6 user stories). Assess the quality of the specification document. Identify issues in the specification document (e.g., inconsistencies within the specification, missing or incorrect functionality, unnecessary functionality ('gold plating'), unclear/umbiguous statements, spelling errors, and so on). Create a well-organised list of all the issues you spot. Each issue should have an ID, a brief description, a remark that helps localise the issue, the type of issue, the severity of the issue. Clearly define the categories 'type' and 'severity' and explain the underlying rationale for the definition of types and severity levels.

   Important: Make sure to document your work – you will have to include it in the lab report!
3. To get maximum marks for Part A, you need to find at least 8 issues and you have to provide correctly all the information requested in point 2 above.
4. Before you proceed to Part B, show your work to the lab supervisor. If you don't do this, you will lose 1 mark for either not attending the lab or for not working on Part A during the lab time.

Part B: Work in Pairs

5. Before you leave the lab, find a partner. Note: If the number of lab attendees is odd, there can be one group of 3 students.

   Important note: If you do not form a pair (or a group of 3), you will not be able to get marks for Part B.
6. Share your individual issue lists and create a consolidated list. This might require that you have to redefine your type categories and severity classes. Explain the reasoning that made you agree on the new definitions. The consolidated list should be well-organised and each issue should have an ID. Make sure that traceability to the individual lists is maintained. In particular, the

consolidated list should clearly indicate whether an issue was found by one student only or by both students.

7. In case you have a disagreement on whether an issue detected by one student is the same as the issue of the other student, or whether an issue found by one student (but not the other) is actually an issue, still take a decision on whether to count the issues as separate or identical and make a comment explaining the different viewpoints (and why it was difficult to come to an agreement).
8. Based on the consolidated issue list, i.e., the number of issues detected and the information given on who detected an issue, make an estimate of the number of remaining (i.e., undetected) issues in the specification document. Use any of the capture-recapture model formulas presented during Lecture 4. Show your calculations.

Part C (optional – bonus task): Work in Pairs – in case only one student does the bonus task, say so clearly in your report. In that case, only the student who does the work will get the bonus marks.

Download the Scilab software tool (an open software comparable to Matlab) from the course web or from https://www.scilab.org/. Familiarise yourself with the Scilab files implementing the capture-recapture models M0, Mt, and Mh. Then produce and report estimates for the number of remaining issues (i.e., number of not yet detected issues) with each of the three models. The input shall be the same for each model and shall be generated as follows (pick the case that applies to you):

- If you work in a pair: Take the consolidated issue list of step 6 above and add the issues of a third (virtual) inspector. The third inspector should have found 20% new issues (i.e., the amount of new issues equals the amount of 20% of all unique issues found by inspectors 1 and 2), 50% of the issues only found by inspector 1, 50% of the issues only found by inspector 2, and 50% of the issues found by both inspector 1 and inspector 2 (i.e., duplicates). If necessary, apply rounding.
- If you work alone: Take your list of issues (step 2 above) and add the issues of a second and third (virtual inspector). The second inspector should have found 20% new issues (i.e., the amount of new issues equals 20% of the amount of issues found by inspector 1) and 80% of the issues found by inspector 1. The third inspector should have found 20% new issues (i.e., the amount of new issues equals the amount of 20% of all unique issues found by inspectors 1 and 2), 50% of the issues only found by inspector 1, 50% of the issues only found by inspector 2, and 50% of the issues found by both inspector 1 and inspector 2 (i.e., duplicates). If necessary, apply rounding.
- Discuss which model is most appropriate in the light of the model assumptions.

Read the Appendix for details on Scilab and the capture-recapture model functions.

Reporting: Submit a report consisting of:

- Part A: Report your individual issue lists (incl. explanation of rationale for choice of format and organisation of each individual list). Each of your individual lists should contain at least 8 issues. Thus, if you work as a pair, there must be two individual lists with explanations.
- Part B: Report the consolidated list (incl. explanation of rationale for definition of issue types and severity classes), plus comments on cases where you had difficulties to come to an agreement); report your estimates of remaining issues (show and explain your calculation).
- Part C: Show the input matrix. Report the number of remaining issues predicted by each of the estimation models. Show the interaction with the the SciLab tool when creating your solutions. Discuss whether the underlying assumptions for each of the estimation models are fulfilled.

Submission Deadline: Lab reports must be submitted no later than 7 days after the lab has finished. For example, if your lab takes place on Tuesday, you must submit the lab report no later than the following Monday at 23:59. Late delivery will result in penalty up to 100% (see section 'Marking' below).

Important: If you work in a team, make sure that the names and student IDs of **ALL** team members are clearly stated **in** the report.

Marking:

- Presence in lab: 1 mark – you get this marks only if you show results of Part A to the lab supervisor during the lab.
- Part A: 3 marks per individual list [6 marks total] – will be adjusted in case the lab report is not written by 2 students.
  - You can get full marks per individual list only if you have described at least 8 (actual) issues, and no information is missing
  - If you work alone: the marks for an individual list of a pair is doubled (i.e., so you can get a maximum of 6 marks)
  - If you work as a group of 3 (exceptional case!): the marks for an individual list are 2/3 of the marks for an individual list of a pair (i.e., so the group of 3 will get at most 6 marks)
- Part B: 3 marks per consolidated list & estimation [3 marks total] – you won't be able to get these marks, if you work alone.
  - You can get full marks only if the consolidated list is complete and full traceability to the individual lists is provided [2 marks] and you make a correct estimate of remaining issues (using a capture-recapture model and showing your calculations) [1 mark]
- Part C (bonus): 0.5 marks for showing the input matrix; 1 mark per estimation using models M0, Mt, and Mh; 0.5 marks for discussion of underlying assumptions [4 marks total]
- Penalties for late delivery: If the lab report is submitted up to 24 hours late, a 50% penalty applies. If the lab report is submitted more than 24 hours late, a 100% penalty applies and the report will not be looked at (i.e., no marks for the report and no feedback on this lab will be given).

Requirements

The following list of user stories (US) have been received from a customer representative (i.e., marketing):

- US1: As a customer, I would like to be able to search for flights, hotels, rental cars, cruises, and packages that combine flights with hotels and car rentals
- US2: As a customer, I would like to search for one-way, return, and multiple-leg flights
- US3: As a customer, I would like to choose the classes/categories of my flights, hotels, and cars
- US4: As a customer, I would like to search not only for myself but for my whole family
- US5: As a customer, I would like to search for nonstop and refundable flights
- US6: As a customer, I would like to restrict my searches to specific airlines, hotel chains, and car rental companies

Note: The following specification should only be checked against this set of requirements (i.e., the 6 user stories listed above).

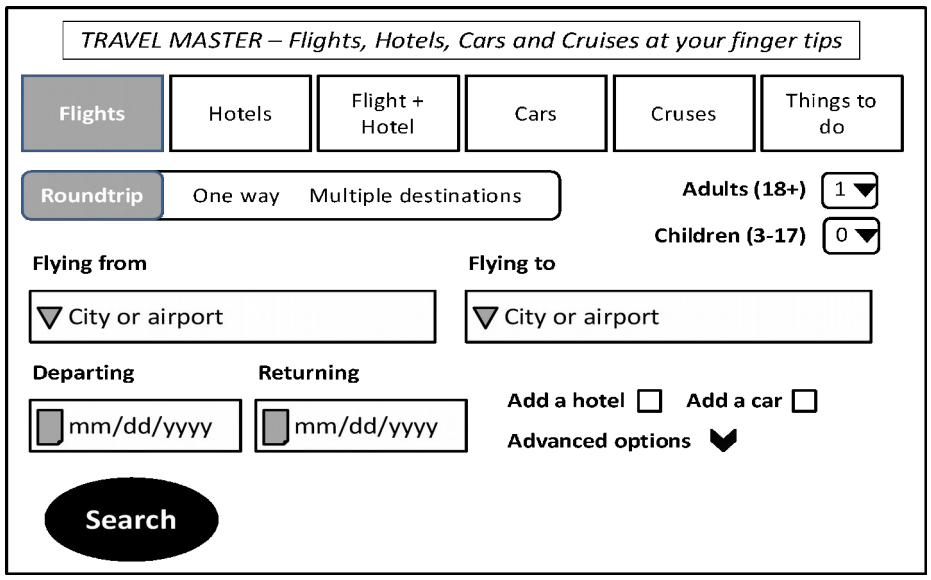
Specification

The actual specification document to be reviewed starts on the next page. Before you start with your review, read the following notes:

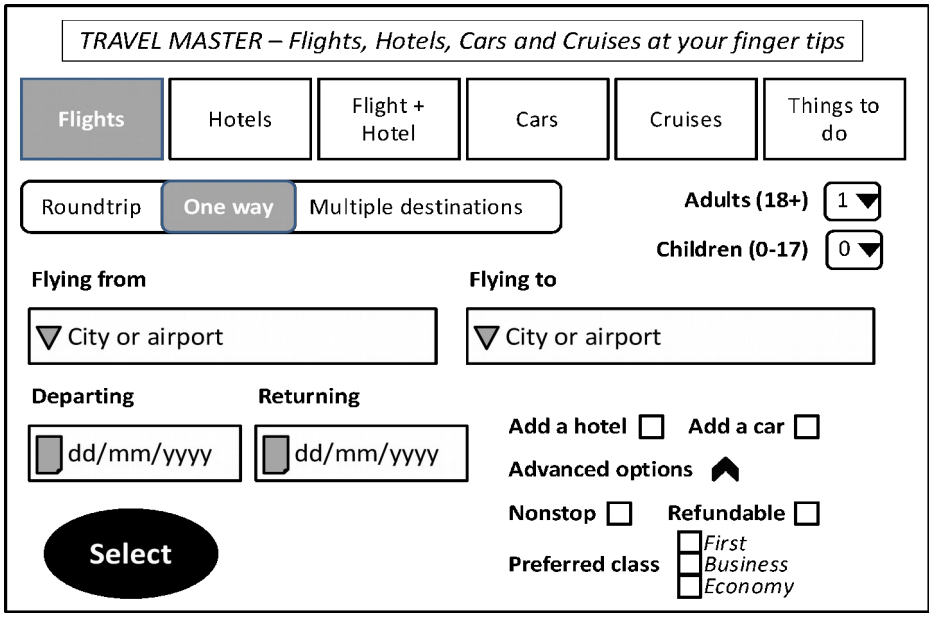
- The specification presented on the next two pages is supposed to correspond exactly with the explicit (and implicit) requrirements listed in section 'Requirements' above.
- The specification consists of both a specification text and two mock-ups of search screens that the web application to be developed will provide. The functionality contained in the two mock-up search screens (and the related textual description) is supposed to address the needs of users (i.e., future customers) that want to search for flight offers.
- Screen 1 shows what a customer sees when selecting search option 'Flights' + 'Roundtrip'. When you review the screen and the corresponding specification text, you should restrict your review to the functionality that should be provided, if a customer wants to search for roundtrip flight offers. Note also that 'Advanced options' has not been selected. You can assume that the customer will see exactly the same information that is shown in Screen 2, if 'Advanced options' had been selected.
- Screen 2 shows what a customer sees when selecting search option 'Flights' + 'One way'. Thus, when you review the screen and the corresponding specification text, you should restrict your review to the functionality that should be provided, if a customer wants to search for one-way flight offers. Note also that 'Advanced options' has been selected. You can assume that the customer will see exactly the same information that is shown in Screen 1, if 'Advanced options' had not been selected.

Specification Document (relating to the 6 user stories listed in the Requirements section):

Screen 1: Mock-up screen in the state after 'Flights' button and 'Roundtrip' button have been selected, and before any additional data has been entered or functiones have been activated by the customer (i.e., user of the web-application to be developed)



Screen 2: Mock-up screen in the state after 'Flights' button and 'One way' button as well as 'Advanced options' have been selected, and before any additional data has been entered or functiones have been activated by the customer (i.e., user of the web-application to be developed)



1. The search screen shows a welcome line with text in the top line
2. The user can select 6 main functions by pressing any of the buttons ‘Flights’, ‘Hotels’, ‘Flights + Hotels’, ‘Cars’, ‘Cruises’, and ‘Things to do’
3. When ‘Fights’ has been selected, the user can select 3 search modes by pressing any of the buttons ‘Roundtrip’, ‘One way’, and ‘Multiple destinations’

   <Note: the following assumes that the ‘Flight’ button has been selected>

4. When ‘Roundtrip’ has been selected (Screen 1), the user can do the following:

   a) Specify flight start location (city or airport) and flight destination (city or airport); this is supported by a pull-down menu (not shown in detail in the screen mock-ups)

   b) Departure date and return date (format: dd/mm/yyyy); this is supported by a calendar menu from where the user can pick the dates (not shown in detail in the screen mock-ups) – alternatively the dates can be entered by the user directly in the specified format

5. When ‘One way’ has been selected (Screen 2), the user can do the following:

   a) Specify flight start location (city or airport) and flight destination (city or airport); this is supported by a pull-down menu (not shown in detail in the screen mock-ups)

   b) Departure date (format: dd/mm/yyyy); this is supported by a calendar menu from where the user can pick the dates (not shown in detail in the screen mock-ups) – alternatively the date can be entered by the user directly in the specified format

6. The user can select the number of adults and the number of childeren (supported by pull-down menues); the default selection for adults is ‘1’, that for children is ‘0’
7. The user can check boxes ‘Add a hotel’ and ‘Add a car’, if offers for a hotel and/or a rental car should be added to the flight offers
8. If the user selects ‘Advanced options’, then he can restrict his flight search to ‘Non-stop’ and/or ‘Refundable’ flights. In addition, he can select the flight class (first – business – premium economy – economy)
9. Pressing the ‘Search’ button will start the retrieval of flight offers
10. Pressing the ‘Check’ button will start the retrieval of flight offers and check the availability of free seats

Note: The specification of *output screens* is NOT shown. Therefore, you cannot know whether they are specified correcly, and thus you should not speculate about any issues related to them. You shall focus exclusively on reviewing whether the specification document actually provided contains any issues related to searches for roundtrip and one way flights in the to-be-developed web application. Also, you should check whether the specification document contains functionality that has not been mentioned (explicitly or implicitly) in the list of requirements and thus would be ‘gold plating’.

Appendix A – 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 issues.
2. Reviewers must not reveal their proposed issues to other reviewers.
3. Reviewers must ensure that they accurately record and document every issue they find. Additionally, the inspection process, for example, at the collection meeting, must not discard any correct issues.
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 issues. Depending on the degree of freedom of the ability of the reviewers and probability of the issues to be detected, four basic models are formed:
    - M0, all issues have equal detection probability, all reviewers have equal detection ability.
    - Mt, all issues have equal detection probability, reviewers may have dierent detection abilities.
    - Mh, issues may have dierent detection probabilities, all reviewers have equal detection ability.
    - Mth, issues may have dierent detection probabilities, reviewers may have dierent detection abilities.

Appendix B – Scilab files

m0mle.sce -- Estimates a point estimation of the issue content using the model M0

Input: Inspection data represented in a matrix of ones (found issue) and zeroes (not found issue).

Row_i = issue no. i, and $column_j$ = reviewer no. j. An example of a matrix is shown below (3 reviewers and 7 unique issues 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 issues.

mtmle.sce -- Estimates a point estimation of the issue content using the model Mt.

Input: The same input as m0mle.sce

Output: The same output as m0mle.sce

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

Input: The same input as m0mle.sce

Output: An array consisting of [estimated number of issues, 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. You load a scilab function by selecting the file containing the model function (e.g., m0mle.sce) in the 'File Browser' window, and then right clicking and selecting 'Execute in Scilab' from the context menue.

You execute a loaded model function by typing the function name of the corresponding scilab file together with the correct data file D. The command would look like this:

-->m0mle(D)

In the example above, D is the required data file containing the inspection data. The data from a text file can be loaded into data file D using the fscanfMat() function, like in the example below:

-->D = fscanfMat('2Rev.txt')