

Exam for ETS200 Software Testing
Lund University, Department of Computer Science
Time: 2016-03-14, 08:00–13:00
Place: Vic:3A-B

Assessment: total 60 points, at least 30 points is required in order to pass the exam.

Answers should be written in English.

Start answering each new task on a new page.

-
1. Define the following terms (one sentence each): (6p)
 - a) Quality
 - b) Regression testing
 - c) Mutation testing
 - d) Integration testing
 - e) Test level
 - f) Test phase
 2. The book distinguishes between four types of defects: i) Requirement/Specification, ii) Design, iii) Coding, and iv) Testing defects. Which type of defects are best found with *white box* and/or *black box* testing methods, respectively? Motivate your answers. (6p)
 3. For the method `bubble_srt` below,
 - (a) draw the control flow graph and calculate the McCabe Cyclomatic Complexity, (3p)
 - (b) define what the McCabe Cyclomatic Complexity means, and what it can be used for in test planning, (2p)
 - (c) define the *minimal* set of test cases needed to achieve 100% decision coverage, (3p for correct answer, reduction for too many test cases)
 - (d) set up def-use tables for the four internal variables in the `bubble_srt` method `i`, `k`, `m`, `n`, and define minimum test cases needed to achieve 100% def-use coverage. (4p)

Note that test cases should include both input values and expected output (arrays).

```
1 public class MyBubbleSort {
2
3     public static void bubble_srt(int array[]) {
4         int n = array.length;
5         int k;
6         for (int m = n; m >= 0; m--) {
7             for (int i = 0; i < m - 1; i++) {
8                 k = i + 1;
9                 if (array[i] > array[k]) {
10                     swapNumbers(i, k, array);
11                 }
12             }
13         }
14     }
15
16     private static void swapNumbers(int i, int j, int[] array) {
17         int temp;
18         temp = array[i];
19         array[i] = array[j];
20         array[j] = temp;
21     }
22 }
```

(Cont'd on next page)

4. A test manager has set as a criterion that their unit testing should reach 95% decision coverage in unit testing. Discuss advantages and disadvantages of having such thresholds. Specifically address if there are differences between application domains. (4p)

5. Industry is very interested in test automation to save costs and increase speed of testing.
 - a) Define benefits and drawbacks of test automation. (4p)
 - b) What characterizes a software project which likely would benefit from automating the test execution? (2p)
 - c) The evolution of test automation can be described in five steps, from basic to advanced:
 - i) Recorded scripts (Capture/replay)
 - ii) Engineered scripts
 - iii) Data-driven testing
 - iv) Keyword-driven testing
 - v) Model-based testing
 Describe each step, and comment the cost/benefit for each of them. (5p)

6. The structural organization of a company is one way to impact on how the work is conducted.
 - (a) Discuss the advantages and disadvantages of the three different approaches (i, ii, and iii below) to organize testers in a company:
 - i) Developers test their own artifacts.
 - ii) Testers are in a separate department.
 - iii) Users are invited to be beta-testers.
 For each of the approaches, assess the effect of the organization on *test effectiveness*, *test competence*, and *test management*. (6p)
 - (b) Suggest approaches to organize a test group for a medium-sized company (30 developers) which develops a mobile app for sales applications, with two native versions for Android and iOS a database back-end connection. Motivate why you choose that approach. (4p)

7. Runeson et al synthesized 12 empirical studies comparing testing and inspection and summarized the findings in Table 3 of their paper (see below). There were nine experiments on code, one in design, and two case studies. Effectiveness (percentage of defects found) and efficiency (defects found per time unit) are presented in Table 3, as well as whether testing and inspection are observed to find different faults or not.
 - (a) Which conclusions would you draw on testing vs. inspection from Runeson et al's summary with respect to *effectiveness* and *efficiency*? Specifically comment the validity of the findings in *experiments* vs *case studies*. (4p)
 - (b) At stackexchange.com, there are different opinions about testing and inspection. Below are three quotes from a discussion thread.
 Discuss pro's and cons of testing and inspection based on Runeson et al's summary and the stackexchange posts. Specifically, assess which sources of information are trustworthy, and why. Summarize the discussion with a recommendation whether a company in the domain of surveillance cameras should apply inspections or not, and how it should be applied, if you advice them to use inspections. (7p)

Table 3**Average values of effectiveness and efficiency for defect detection**

Experiments	Code	Study	Inspection effectiveness (%) ^{*,†}	Inspection efficiency ^{†,‡}	Testing effectiveness ^{*,§}	Testing efficiency ^{†,§}	Different faults found
		Hetzel ¹⁷	37.3	–	47.7; 46.7	–	–
		Myers ¹⁸	38.0	0.8	30.0; 36.0	1.62; 2.07	Yes
		Basili and Selby ²	54.1	Dependent on software type	54.6; 41.2	–	Yes
		Kamsties and Lott ¹⁹	43.5 50.3	2.11 1.52	47.5; 47.4 60.7; 52.8	4.69; 2.92 3.07; 1.92	Partly (for some types)
		Roper et al. ²⁰	32.1	1.06	55.2; 57.5	2.47; 2.20	Yes
		Laitenberger ¹¹	38	–	9 [#]	–	No
		So et al. ¹²	17.9, 34.6	0.16; 0.26	43.0	0.034	Yes
		Runeson and Andrews ¹³	27.5	1.49	37.5	1.8	Yes
		Juristo and Vegas ¹⁴	20.0 –	– –	37.7; 35.5 75.8; 71.4	– –	Partly (for some types)
	Design	Andersson et al. ¹⁵	53.5	5.05	41.8	2.78	Yes for one version, no for the other
		Conradi et al. ¹⁶	–	0.82	–	0.013	–
		Berling and Thelin ³	86.5 (estimated)	0.68 (0.13)	80	0.10	Yes

From stackexchange: What should come first: testing or code review?

Ideally, in an Agile world, both :)

Test-driven development is a method which encourages the development of unit tests prior to the writing of actual code - this way, you can capture the specification in code and write tests that pass the tests. Following that, automated integration tests that ensure all the various different components fit together are a Good Thing to further ensure the functionality of your application matches what is expected.

As for code reviews, pair programming is a useful way of having another mind overlooking your code as you're actually writing it. However, that's not necessarily a practical approach. The way it works in my current company is that code is reviewed after it has been tested on the developer's personal machine, but before it has been deployed to a shared development server.

Code review is done to "polish" things that already work, to assure the code has the desired quality level and meets the code guidelines defined by the company.

Code review can also happen as part of the future general optimization activity where you refactor and improve the old code.

If you practice code review before doing a check-in then code review falls between two testing stages: you as a developer test your code first, your peer does code review, you check it in, then later dedicated testers will perform more thorough individual and integrations tests.

Test first. Test last. Test, test, test.

Code review is nice-to-have. But difficult - can be a painful process if personalities involved or differing opinions.

Testing is very clear: either it works or it doesn't work. So test, test, test! And code-review if possible.

(Last page)