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.

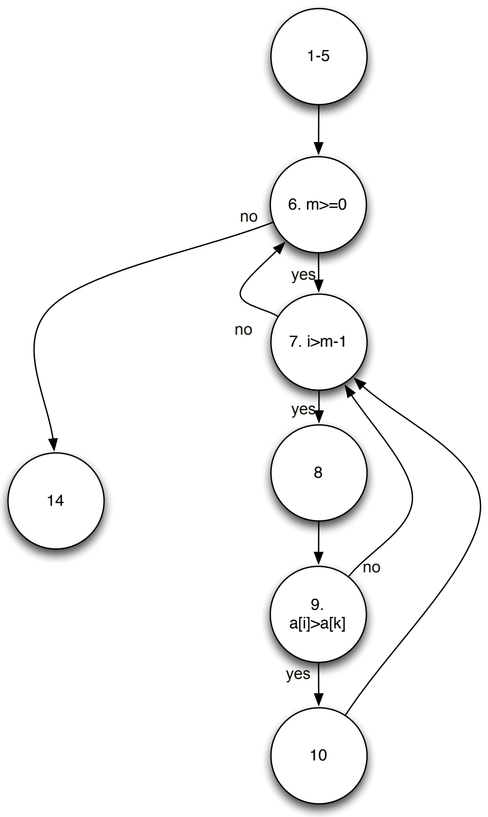
---

1. Define the following terms (one sentence each): (6p)
   1p per correct definition. 0.5 p if some key aspect of the definition is left out.
   a) Quality
      The degree to which a system... meets requirements, or the degree to which a system meets user needs.
   b) Regression testing
      The retesting of software to demonstrate that previously working software is not damaged by other changes.
   c) Mutation testing
      A method used to analyze weather test cases are good enough, by changing (mutating) program concepts and then running test cases to assess their ability to detect the changes (kill mutants).
   d) Integration testing
      Testing aimed to detect defects on the interfaces between units.
   e) Test level
      A group of test activities that focus into certain level of the test target.
   f) Test phase
      A test activity, bound in time*. Does sometimes match the test levels. In iterative development the phases may not match the test levels. Phases might match the test levels as depicted in V-model.*

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)
   i) *Requirement/Specification – only BB testing, since only BB takes the high level view of the system.*
   ii) *Design – Could be both. Either design defects are found the WB test are generated from the code structure, or BB tests are generated from the specifications.*
   iii) *Coding – WB only, since it only focuses on that detailed level.*
   iv) *Testing defects – WB when executed or reviews if that is applied to test code.*
   0,5 p for answer, 1p for motivation

3. For the method `bubble_srt` below,

(a) draw the control flow graph and calculate the McCabe Cyclomatic Complexity, (3p)



$MCC = 9 - 7 + 2 = 4$ *or* $3 + 1 = 4$

(b) define what the McCabe Cyclomatic Complexity means, and what it can be used for in test planning, (2p)
MCC is the number of independent *paths through the graph. Note that is its* not *equal to path coverage. MCC can be used as an approximation of the numer of test cases, but not an exact foracast.*

(c) define the *minimal* set of test cases needed to achieve 100% decision coverage, (3p for correct answer, reduction for too many test cases)
One test case with the minimum or maximum number in the middle element.
0p for more than one test case. Deduct 1p if the sorted array is not presented as expected output.

(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)
Def-use pairs:
i: 7-7 7-8 7-9 7-10
k: 8-9 8-10
m 6-6 6-7
n 4-6
1p per correct variable, -1 if swapNumbers is included

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++) {
```

```
               k = i + 1;
               if (array[i] > array[k]) {
                    swapNumbers(i, k, array);
               }
           }
       }
  }

     private static void swapNumbers(int i, int j, int[] array) {
        int temp;
        temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
}
```

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 adress if there are differences between application domains. (4p)

   Advantages:
   Measurable criteria that has some *value*
   Can give an overview of test status
   Guide for design of white box tests
   Disadvantages:
   Risk for using criterion as Key Performance Index
   Impossible to handle e.g. dead code
   Can rely too much on criterion and give false security
   Domains:
   May be mandatory in some domains (safety)
   0.5 p per advantage/disadvantage (up to 3p), 1p for comment on domain

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)
      +repeated testing at minimal cost
      +enables massive test execution (e.g. after each build)
      +quick check of change impact/regression testing
      - high costs for investment in automation
      - high costs for maintenance of tests
      - lack of variation in tests/lower effectiveness
      1p per motivated benefit/drawback, 0.5 p without motivation
   b) What characterizes a software project which likely would benefit from automating the test execution? (2p)
      Many iterations, Few changes to interfaces, 1p each
   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)

   i) Recorded scripts (Capture/replay) – Easy to capture hard to maintain
   ii) Engineered scripts – e.g jUint, takes more to design, more maintainable
   iii) Data-driven testing – Separate logic and data, separates concerns (data vs flow)

iv) *Keyword-driven testing – Domain Specific Language for testing. Reduces dependency on interface specification*

v) *Model-based testing – Good for small, well defined domains. Expensive to build model*

0.5 p per description, 0.5 p for cost/benefit

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.
      Pro: well known, quick feedback,
      Con: bias of knowing the software too well, dual competency need for developer and management
      ii) Testers are in a separate department.
      Pro: specialists, both for testers, managers and test competency and tools,
      Con: distance wrt communication.
      iii) Users are invited to be beta-testers.
      Pro: cheap and realistic,
      Con: risky, no control

   For each of the approaches, assess the effect of the organization on *test effectiveness*, *test competence*, and *test management.*
   *2*3*0.5 p for each reasonably complete answer on pro/con. 0p for bullet items. 1p for motivation related to user. 1p for motivation related to product line. 1p for motivation related to size*
   (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 for Android and iOS, and a database back-end connection. Motivate why you choose that approach. (4p)
   For example:
   On the development side: specialists for each technology (IOS, Android, Database)
   On the testing side, utilize similarity between apps, max two groups.
   Testing in dev teams and for whole systems
   2p for general description with motivation.
   1p for reference to application domain.
   1p for combining iOS and Android test

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)

   Testing more efficient and effective - 1p
   More nuances in the answer on type of testing - 1p
   More nuances in the answer on type of study -1p
   Difference between efficiency and effectiveness - 1p

   (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)

Discussion on general pro+con, referring to both Runeson and stackexchange -2 p
Assess strength of evidence - 1p
Question test vs inspection definition. Do we know what is compared in each case? - 1p
Discussion on specific domain solution (surveillance cameras) - 2p
Referring to guest lecture (Axis) -1p

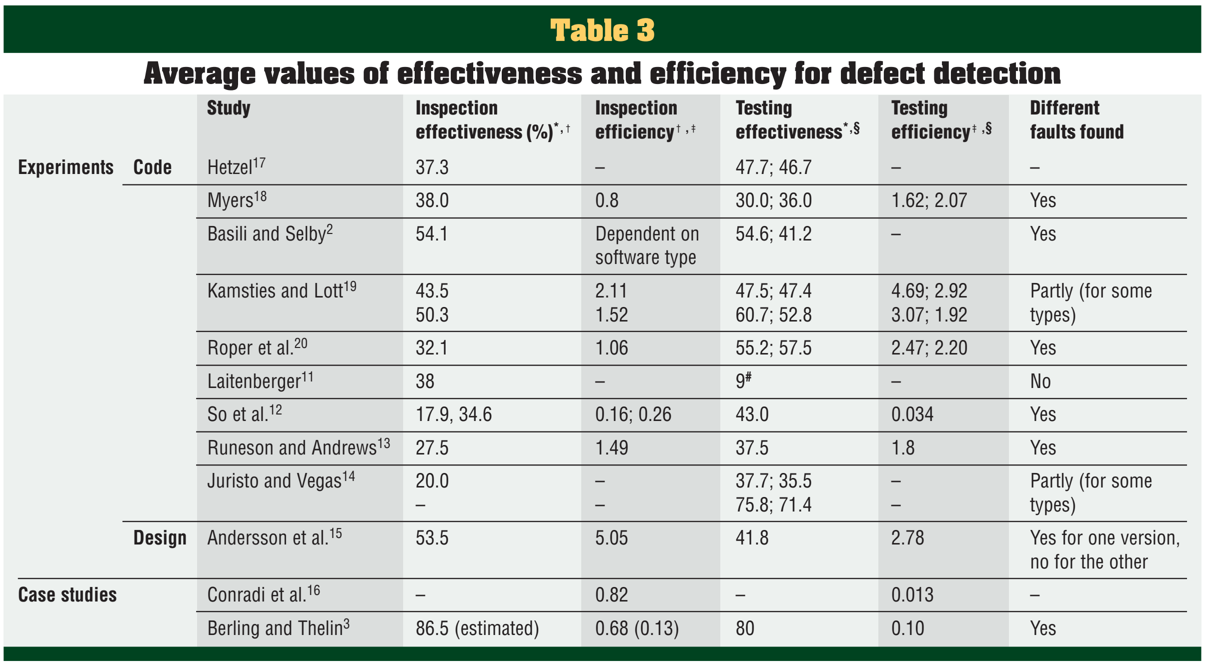


Table 3

Average values of effectiveness and efficiency for defect detection

		Study	Inspection effectiveness (%)[*, †]	Inspection efficiency[†, ‡]	Testing effectiveness[*, §]	Testing efficiency[‡, §]	Different faults found
Experiments	Code	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
Case studies		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)