Coursework commentaries 2015–16

CO2226 Software engineering, algorithm design and analysis

Coursework assignment 1 – Testing android applications

General remarks

This commentary explains the difference between the various levels of performance for the set of coursework CO2226 (Coursework 1). The commentary does this by drawing out the general principles that apply to all courseworks and then illustrating these with examples of good and poor answers.

As in previous years, you are reminded that you should not approach the examination in terms of seeking a **unique** 'correct answer'. Rather, you should seek to show that you have a deep understanding of the issues that combined render software development one of the most challenging, but potentially exciting and rewarding, of all human activities.

In the case of the current coursework, as in previous years, a software engineering scenario should be considered. You are presented with a challenge: a problem concerning android testing. Mobile applications are increasingly important to software engineers because of their performance, and testing remains a critical underpinning quality-assurance technique. The android platform has retained a significant market share, making automated android testing an important technical and scientific challenge.

As with the previous years' coursework, a large portion of the marks relates to black-box testing, one of the most prevalent and important software quality assurance techniques used by software engineering practitioners. This is complemented by a more research-focused investigation of possible future developments in automated testing.

Unsatisfactory answers either failed to address the question or 'padded out' an otherwise short and weak answer with 'book work' that was not sufficiently related to the question. This has been commented on in previous reports on this coursework.

You are reminded that inclusion of irrelevant material reduces the overall mark awarded, and therefore you are well advised to refrain from this practice. Some students simply did not answer the question fully and this also resulted in low marks. As a general rule, the advice this year remains the same as in previous years, that is, you should ask yourself two questions:

1. Is my answer too short? That is, I should check that I have covered everything and supported my answer **with explanations** and relevant details.
2. Is my answer too long? That is, I should check that I have not included **irrelevant** information and 'book work' material that does not address the question I was asked but just gives generic information that is related to the topic of the question but not the substance of the required answer.

Pass level answers tended to give correct, but somewhat generic answers. They tended to fall into the trap of including material that was only tangentially relevant rather than directly relevant. Such answers were typically correct but vague. This could be equally true of almost any scenario. Such answers failed to adequately account for the specifics of the scenario.

Good answers showed an appreciation of some of the deeper issues that may affect a software project like the one described (note that these can be either non-technical issues or issues that involve both technical and non-technical issues). In order to do this, you needed to carefully consider the scenario and to think more deeply about the issues that may affect the manner in which your technical work on the project would proceed. This is a very valuable skill for any engineer and it is particularly important for software engineers. The various components of information given in the scenario are not all equally important; some have a direct bearing on the technical issues, while others are more nuanced.

An excellent answer was one that was able to draw on wider reading to discover algorithms and techniques that could then be applied to the problem presented in the coursework.

The guidelines for pass level and good answers remain substantially unchanged from previous years. The guideline for excellent answers is specifically pertinent to questions where you were asked to research and apply advanced software engineering techniques. We shall now consider how they apply to the specific two tasks for coursework 1 for the year 2015–16.

Comments on specific questions

Question 1

Answers to Task 1 were generally good. Students were able to define a reasonable set of black-box test cases, and demonstrated awareness of android-specific and mobile-platform-specific testing requirements. The number of test cases varied dramatically but the primary determinant of the overall mark was the justification and explanation of the test cases, as with previous years' coursework. Having said that, testing is about catching bugs and more thorough test suites do tend to catch more. Hence, a correlation, although not a causal relationship, was apparent between students who tended to produce a lot of tests and the higher overall marks.

As in previous years, no matter how many test cases were offered, it was insufficient to simply list the set of black-box test cases. Students who did this had very low, often unsatisfactory, marks. It is very important to explain the reason **why** each test case was included.

Also, students are reminded that testing is neither only nor exclusively about finding bugs, but is about the trade-off between the test effort and the chance of fault revelation. Testing incurs effort, and hence, although increased numbers of test cases can improve effectiveness, it reduces efficiency if it fails to improve the effectiveness; testing the same thing twice (in slightly different ways) is far less effective and efficient than testing two different aspects of the system.

This means that an excellent software engineer will seek to achieve the maximum achievement of testing goals with the tightest budget of test effort. Of course, testing needs to be adequate, and thus, below a certain number of test cases, inadequacy of the overall testing process can also lead to failure on this kind of coursework question.

Question 2

Task 2 concerned search-based approaches to software testing, which are increasingly prevalent in the research literature and software practice (particularly in the area of testing). Similarly to last year, responses to Task 2 tended to be less well answered, on the whole, because many students simply researched the content from the internet and included it, largely verbatim, in their answers. Students are reminded that this form of answer, even with suitable quotation and referencing, is adequate for only a bare pass but is insufficient for a higher grade. Furthermore, in the absence of proper quotation and citation, this would be classed as plagiarism; it was reassuring that the vast majority of students did include proper quotation and referencing.

Similarly to previous years, students who performed better on this task related the answers found in their research concerning the nature of Search Based Software Testing (SBST) to the specific problem in hand, explaining how it could be useful in testing android. Furthermore, students who obtained the highest marks demonstrated that they were able to apply these techniques, computing the effect of the search-based algorithms by hand, on the specific set of test data produced.

The best answers tended to contain an original worked example which demonstrated a deeper understanding of the underlying concepts and principles, by demonstrating their practical application.