Coursework commentaries 2016–17

CO2226 Software engineering, algorithm design and analysis

Coursework assignment 1

General remarks

This Software engineering coursework required students to provide some analysis and black-box test cases for a given scenario in a software engineering context. The work required followed a similar pattern to previous years, but with a different scenario and context, and an extra task of explicitly identifying stakeholders.

The Examiners are looking for a demonstration of a deep understanding of the issues relevant to software development, rather than identification of a single solution that is perceived to be the correct one. There will be many relevant aspects, and these should all be brought together in a coherent way. In the case of this coursework, as in previous years, a software engineering scenario was presented: a problem related to providing a software system to manage specific aspects of a medium-sized building contractor company.

Additional aspects of the context included setting the questions from the perspective of the software development company, the varied customers of the building contractor company, and the variety of tasks related to the building contractor company's day-to-day operation. In addition, a test-driven approach to the development was specified, requiring you to understand and use a specific software development approach. To complete the coursework, you also needed to assimilate some background reading on Test case prioritisation.

In responding to the tasks, students were required to firstly identify the stakeholders, and the relationships between them. Explicit identification of stakeholders is a very important part of any context analysis. As with the previous year's coursework, a large portion of the marks related to black-box testing, one of the most prevalent and important software quality assurance techniques used by software engineering practitioners. This was complemented by a more research-focused investigation of factors related to testing a project that is expanding, and use of the TCP techniques.

There were some submissions that did not include answers to all three tasks, and these of course obtained lower marks. Aside from this, answers were generally reasonable, with some good or excellent ones. Most candidates obtained at least a pass mark, except for those who had missing sections (or those who were referred for plagiarism).

Unsatisfactory answers either failed to address the question or included material 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 strongly advised to refrain from this practice. Some students simply did not answer the question fully and this also resulted in lower marks. As a general rule, the advice this year remains the same as in previous years.

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 that does not address the question 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 tend to give correct, but somewhat generic answers, presenting material that is only tangentially relevant rather than directly relevant. Such answers fail to account for the specifics of the scenario adequately.

Good answers show an appreciation of some of the deeper issues that may affect a software project like the one described (note that these may be non-technical issues or combine both technical and non-technical issues). To do this, you need to consider the scenario carefully, and think deeply about the issues that might affect the way 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.

Excellent answers will draw on wider reading, for example to discover algorithms and techniques that might be applied to the problem presented in the coursework.

Comments on specific questions

Question 1

This question was generally reasonably answered. Some students considered the stakeholders in the entire scenario — that is, those working for UCL as well as those employed by Gareeno-Leake, and also those who would be affected by the system being developed, such as customers of the company. The task specification was not explicit on this, so such answers were considered complete. However, most students focused on stakeholders impacted by the software system only, and included customers and employees of Gareeno-Leake. Some also included suppliers to the building company, who would, of course, be impacted by the stock and finance aspects of the system. A small number of answers stated explicitly which aspects of the scenario were being considered for the stakeholder analysis; it is always a good idea to do this.

Question 2

Answers to this question were generally good. Students were able to define a reasonable set of black-box test cases, and showed awareness of testing requirements for a system managing a variety of office and business functions. The number of test cases varied but the key coverage was the justification and explanation of the test cases, as with previous years' coursework. Having said that, testing is in part 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. You are again reminded that testing is not only about finding

bugs, but is about the trade-off between the test effort and the chance of fault revelation. Testing incurs effort, and while 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. An excellent software engineer will try to achieve the maximum achievement of testing goals with the tightest budget of test effort. That said, testing must be adequate, and with too few test cases, inadequacy of the overall testing process will lead to unsatisfactory marks on this kind of coursework question.

Question 3

This question concerned Test case prioritisation which aims to reduce the amount of re-execution of test cases, by attempting to identify the most faults in the tests of higher priority. As in previous years, responses to this task — which was a research one — tended to be less well answered on the whole, because many students simply looked for content from the internet and included it, largely verbatim, in their answers. You are reminded that this form of answer, with suitable quotation, citation and referencing, may just prove adequate for 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 most students included careful acknowledgement of sources. As in previous years, students who performed well on this task related the answers found in their research concerning TCP to the specific problem in hand, explaining how it might be useful in optimising the testing of an enterprise software system. Furthermore, students who obtained the highest marks demonstrated that they were able to apply these techniques 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.

Coursework commentaries 2016–17

CO2226 Software engineering, algorithm design and analysis

Coursework assignment 2

General remarks

This coursework was a programming exercise where students were asked to answer seven questions. The questions were based on the implementation of Dijkstra's algorithm for the shortest path problem, and fragments of code were provided to be used in the context of the capital cities problem. You were expected to modify the code supplied and add your own code to answer the questions. Some questions were based on previous ones, and the two main criteria (apart from, obviously, correctness) were reusability of data structure components (so that the same calculations do not have to be made from scratch and the existing calculations can be reused to the maximum level) and speed of execution.

Common mistakes identified in the submissions are listed below; note that these mistakes were penalised by loss of marks, rather than a flat zero.

- Where IDEs are used (e.g. NetBeans, Eclipse, IntelliJ) a package statement is inserted at the beginning of the code. This line had to be removed, to avoid a compilation error.
- The use of IDEs also imposes a structure, for example code files in a directory called src, data in a directory called data, libraries in a directory called lib and so on. This broke the assumption that all data files about countries, connection data and city geographical data should be in the same directory as the main file. This required editing to remove the prefix src so that all files can be found in the same directory.

Below is a list of the common mistakes identified in the submissions; please note that these mistakes were penalised with marks deducted from the total marks allocated for each question.

Some students:

- Put all three files to be read from the command line, and some submissions even required specific extensions. Again, it is an easy fix to meet the coursework specification – two files without extensions should be read from the command line;
- Hard-coded the country ids for the questions – finding them from the text file provided was part of the question;
- Included incorrect and/or location-dependent paths for files; a few submissions had hard-coded file paths that included the local hard disk;
- Included incorrect Java statements, for example spelling mistakes, such as Sytem.out.println;
- Enclosed arguments used as variables in double quotes. So, for example, using "args[1]" as an argument for the FileReader class, then the program looks for a file called args[1] rather than the file passed from the command line.

- In some cases, although the code ran without problems, the order of command-line arguments was reversed; this code required inspection to see what was expected. Make sure your code runs as specified in the brief.
- Some submissions had the main method in the Graph class which, although not wrong technically, fails to follow the instructions in the coursework brief and adds the main method for a project in an auxiliary, essentially helper, class. This doesn't cause issues with the running of the programme per se, but where there is a main class and an inner class, it would be expected that the main method would be in the main class, not the inner class.
- Some had classes where an import package statement was missing, so they could not be recognised, or where they were defined in another file so the current program could not recognise them. Do make sure that all classes you are using in your coding exist (e.g. are either Java built-in or you have already defined them).
- In the question for cities with the minimal number of vertices, some of the answers provided included a pair of cities with direct connections. Please read the coursework brief carefully – it was clearly stated that shortest paths with a length of one should be discarded.
- Another common mistake was to forget that inner classes were required (in this case the class Graph). If this class was missed out, the program would not compile as the constructor for the Graph class is used in the programme. This means that Examiners could not run the program, resulting in zero marks.

In general, you should first make sure that your code compiles without errors – pay close attention to any error messages you are getting and understand why you are getting them, and what you should do to get rid of them. If you used an IDE for development, please make sure that your programme compiles without errors from the command line before you submit it (the Examiners will only be using the command line to run your programme and rely solely on Java's built-in classes). Also, think very carefully about what data structures you should be using and why. You should put effort into understanding what information is required to answer the questions and how you can reuse this information; a program that recomputes information that it had already computed is slower, less effective, harder to debug, and nonintuitive.