Coursework commentary 2017–18

CO2226 Software engineering, algorithm design

Coursework assignment 1

General remarks

This commentary provides general feedback on the questions and the performance of students for the first coursework assignment of the CO2226 module in 2017–18. A general comment on the answer expected is provided as well as comments on the students' performance and mistakes.

This software engineering coursework assignment was based on a business case scenario (Cabs4All), which described the requirements as well as the different levels of access for the application users. Students were asked to identify the stakeholders in this application as well as how they are related, and to draw a set of test cases for the main functionalities of Cabs4All (i.e. those with critical business importance). The coursework assignment also contained a research element, in which students were asked to research the use of Automated Testing and its applicability for the Cabs4All project.

In general, most students answered the questions well and without problems. The questions were straightforward and easy to understand, and the scenario was detailed enough, providing the necessary information for the process. Students were expected to start from the definition of what they were asked to analyse and then look at the scenario.

Examiners were looking to identify the following three main areas in the submissions:

- evidence of looking beyond functional requirements to identify stakeholders (but not on software and services used as tools)
- the heavily data-centred nature of test cases
- critical analysis in arguments about whether automated testing was a good idea for this application or not.

These were identified, raised and answered in a satisfactory manner, but in most cases the more critical and in-depth view was missing.

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 answered well, although in many cases the relationship between the different stakeholders could have been made clearer by expanding on the brief description. Some students used a number of Unified Modelling Language tools (UML), for example, use cases to identify stakeholders; although this has some validity and will give you a good insight into the answer, it was not part of the question and the use-case model focuses on the functional requirements. A good place to look for potential stakeholders is "soft" requirements (for example, business politics), which the UML completely ignores. Finally, a number of students identified systems used for specific functionality (for example, PayPal or Visa for payments) as stakeholders; they are not, as they do not influence decision-making – they have only been chosen as tools. Obviously, they have an interest in the company doing well, but that does not make them stakeholders.

Question 2

In answer to this question, many students provided too many tests; this is not required, nor requested, and shows difficulties in identifying the key and critical areas. Once the tests are chosen, we have to understand that they need a detailed execution plan and they need to be data-centric. Some students described test cases with instructions to simply open a page and type some data – this is not an adequate specification. If the required information cannot be found, how would you know if this is because it does not exist in the database, or because the program has a bug and cannot find it? How would you expect someone who knows nothing about the system to carry out the steps unless they have been provided with explicit details? How would we measure whether the test is successful or not if we do not know what kind of data to expect as the result? A test case should always have a clear rationale, specific pre-conditions as to what data exists on the system before you run it, a strict execution plan and a measurable outcome. This is where most of the marks were lost. In addition, many students only considered the functional requirements of the system, but business critical systems have non-functional requirements as well; what use is a correct system if it is too slow or insecure? Once again, the definitions of slowness or security will have to be quantified – we run a test and get measurable outcomes, and we need to compare them against a pre-set benchmark.

Question 3

For this question, the main element that examiners are looking for is critical analysis. Many submissions only gave a list of tools and their marketing description from the Internet – a few without even the references. What we are looking for here is, firstly, the testing profile that is needed for the application based on the requirements, its "application profile" and whether or not an approach of Automated Testing will help (or maybe it will help certain facets of the application and not others). Many submissions showed no critical analysis and simply stated that Cabs4All would benefit, without providing any justification. Many students only saw a positive side to Automated Testing, assuming that nothing can go wrong when you apply it. Some pertinent questions are:

- what happens to the different types of requirements (for example, functional versus non-functional) when you use Automated Testing?
- is Automated Testing better suited for some requirements rather than others?
- what pitfalls should the company watch out for, and why?

Coursework commentary 2017–18

CO2226 Software engineering, algorithm design

Coursework assignment 2

General remarks

This report provides a general commentary on the questions and the performance of students for the second coursework assignment of the CO2226 module for the academic year 2017-18. A comment on the answer expected in generic terms will be provided as well as comments on the students' performance and mistakes.

This coursework assignment was a programming exercise where students were expected 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 already provided; these fragments were supposed to be used in the context of the "airports" problem. Students were expected to modify the fragments and add their own code to answer the questions. Some questions were based on previous ones and the two main criteria (apart from, obviously, correctness) were speed of execution and re-usability of data structure components, so that the same calculations do not have to be made from scratch and the existing calculations can be re-used to the maximum level. The result for each question would be an all or nothing mark, depending on whether the result produced by the student's submission was correct or not.

Common Mistakes

The list below is a list of the common mistakes identified in the submissions. Please note that students committing these mistakes did not receive a zero mark but were penalised by having marks reduced from the mark allocated for each question:

- As a result of students using intergrated development environments (IDEs) for writing the code (e.g. NetBeans, Eclipse, IntelliJ, …), the IDE was inserting a 'package' statement at the beginning of the code. This line had to be removed as an error would occur when compiling.
- The use of IDEs (e.g. Eclipse, NetBeans, IntelliJ, etc.) resulted in a structured way of writing code (e.g. code files in a directory called src, data in a directory called data, libraries in a directory called lib, etc.), but broke the assumption when running the file that all data files about countries, connection data and city geographical data should be on the same directory as the main file. This was easy to fix, as all that had to be done was to remove the prefix src so that all files can be found in the same directory.
- The coursework assignment specification clearly stated that you need to read three files from the command line without their extensions. Some candidates put two of the three files to be read from the command line (again, an easy fix but not in line with what the coursework asked for).
- Some submissions included code that appended the extension '.csv' – this is wrong, the files should be called exactly as stated in the assignment brief.

- Some submissions expected additional information and arguments following the filenames (for example, 'airport ids') – the brief was clear in that only the filenames should be provided as arguments and nothing else.
- Some students hard-coded the airport ids for the questions; this is incorrect, extracting them from the text file provided was part of the question.
- Do make sure that the path for the files is correct and location independent – a few submissions had hard-coded file paths that included the local hard disk.
- Make sure that the Java statements are correct – some submissions had spelling mistakes in the Java statements (e.g. Sytem.out.println).
- A number of candidates assumed a bi-directional graph, so in the loop for reading the edges file they would have index i running from 1 to N and index j running from i+1 to N. The graph is directional, so both indices should run from 1 to N.
- Some students assumed that if the 'airport' file has, for example, two hundred entries, then this is the maximum for the index for the structure holding airport ids and linking them to names – this is wrong as the actual index (unless you use a more complicated structure) would use the airport id as the index.
- When you use arguments as variables, you should not put them in double quotes. Some students used "args[1]" as an argument for the FileReader class, and the program looked for a file called args[1] rather than the file passed from the command line.
- Please read the instructions carefully, and follow them; there were a few submissions where, although the code ran without problems, the order of command-line arguments was reversed and it was necessary to examine the actual code to see what was expected rather than run it as specified in the brief.
- Not a problem that will cause issues with the running of the program as such, but if you have a main class and an inner class you would expect the main method to be in the main and not in the inner class. Some students had the main method in the Graph class: although this is not technically wrong, it does not follow the instructions from the coursework brief and adds the main method for a project in an auxiliary, essentially helper, class.
- Do make sure that all classes you use in your coding exist (i.e., they are either Java built-in or you have already defined them). Some submissions had classes for which either an import package statement was missing and they could not be recognised, or they were defined in another file and could not be recognised by the current program.
- In the question for airports with the minimal number of connections, some of the answers provided included a pair of airports with direct connections. Please read the coursework brief carefully, as it was clearly stated that shortest paths with a length of one should be discarded.
- Another common mistake that caused candidates to lose marks was that they forgot 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 program. This means that the examiner cannot run the program at all, resulting in no marks for the student.

Conclusion

In general, you should first make sure that your code compiles without errors – pay close attention to any error messages, 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 program compiles without errors

from the command line before you submit it. The examiners will only be using the command line to run your program, and will 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 an effort into understanding what information is required to answer the questions and how you can re-use this information; a program that re-computes information that it has already computed in the past is slower, ineffective, harder to debug and non-intuitive.

In terms of candidate performance, there were no questions which the vast majority had problems answering. Some of them (e.g. 2 and 5) had multiple correct answers and some of you returned all of them and others one; both answers resulted in full marks as no specific instructions had been given on this one. A number of good techniques were used both in the Shortest Paths as well as the Dijkstra method to avoid direct links, and in most cases they worked without problems (although, when you make changes to the code it is always advisable that you run them through some extreme scenarios and make sure that the values returned are correct). Overall, most candidates using the pseudocode provided achieved high marks, and submitted robust, well structured and well written programs.