Examiners' commentary 2017–2018

CO2226 Software engineering, algorithm design and analysis – Zone B

General remarks

The examination was set generally as a mixture between questions that test the candidate's basic understanding of the material, and questions that require the candidate to apply their knowledge and demonstrate deeper understanding. Candidates are reminded to read each question carefully and address all aspects of the question.

The paper is perhaps unusual in that it covers two separate (though loosely related areas): Software Engineering as well as Algorithms.

This examination was therefore split into two parts. Part A assesses the Software Engineering and Analysis material, Part B the Algorithms material. Each part has three questions, of which two are to be answered (therefore four questions answered in total, two from A and two from B).

Candidates should, however, be sure to number their answers clearly and correctly. There were some cases where it was unclear. Though examiners will do their best, if we cannot read or make sense of it, it makes it very hard to award marks.

Furthermore, if candidates wish an attempt at a question not to be marked, they should indicate this clearly (e.g. by crossing it out). Examiners are directed to mark only the first two questions from each part. There were a few instances of this.

Again, examiners would appreciate it if candidates were to cross rough work through so it is not inadvertently marked.

All questions require a mixture of bookwork (e.g. explaining terms, giving definitions, and so on.) and application (e.g. calculations, building UML models, and so on.) with the majority being application. Candidates should work on practicing likely calculation/application tasks.

For Part A (software engineering), the key application skill is to produce a (usually UML) diagram from a scenario, the same scenario possibly being shared between questions. Candidates should ensure that they can do this well and quickly. We hope that the discussion below will help candidates to be mindful of where marks can be easily missed.

Some brief explanatory comments on answers involving (pseudo)code in Part B would be helpful, as some answers were unclear.

Also, candidates would do well to show working on questions that involve calculation or the application of a process. This applies doubly in part B where a mistake midway can lead to a wrong answer in many questions. Where this has happened often will be highlighted later.

If the candidate simply writes down what they think the answer is and it is wrong, then they will get no marks. If we see working, we can see what the candidates understand and they may gain partial marks.

Candidates should also note that mark schemes often explicitly allocate marks to showing working and giving reasons for their answers (and questions may ask for this).

Comments on specific questions

Question 1

This started with a software verification/validation question where candidates were asked about the relationship between UML sequence and use diagrams in verification. This was followed by the usual "generate a software engineering model from a scenario" task in the domain of an events organisation website (in this case, using UML class diagrams).

Part (a) as noted above was bookwork. Candidates were generally able to explain what the diagrams were (even though this was not asked for). It was when candidates were asked to say how they correspond with each other and how one would verify them that the answers sometimes became vague and rather general.

A few short paragraphs would suffice for a good answer here. Some candidates either gave very terse answers (common) or wrote essays (unusual but wasted time), to their detriment.

In part (b), the quality of class diagrams was variable. There were a number of areas where candidates lost marks (in line with previous years):

- In general, candidates were able to capture the majority of the scenario.
- Some candidates produced diagrams with few classes (5-6 is definitely too few), and/or omitted key classes.
- Sensible proposals were generally given for class attributes and methods.
- More commonly, candidates failed to fully illustrate associations, composition and aggregation relationships between the objects despite these being explicitly requested in the question.

The scenario is designed to give you a chance to show you can use all of the above, so make sure you take it.

The construction of class diagrams seems to be a consistent weakness compared with other UML diagrams. Candidates are losing marks needlessly by not preparing fully for it.

Question 2

Question 2 started with a bookwork question on representing concurrency in an activity diagram, followed by a scenario task (activity diagram, extending the scenario from (1b).

Part (a) was not answered well. A question like this comes up regularly, so candidates should be well prepared. Again, a few paragraphs would suffice here.

Generally, part (b) was answered well, and candidates were able to put most of the scenario on the diagram. Common errors included:

- Some candidates produced diagrams with missing swim lanes.
- More commonly, candidates failed to fully capture concurrency and flow of control between actors in their answers.
- In general, candidates should pay particular attention to identifying activities, decision points and fork/join nodes.

The scenario is designed to give you a chance to show you can use all of the notation, so make sure you take it.

Question 3

Question 3 started with a bookwork question on the concepts of abstraction and interface in Object-Oriented design, and then addressed the development of a state machine diagram based on the same scenario as the previous two questions.

Part (a) was generally not answered well and many answers were vague. There were examples given in the stronger answers and this lifted the quality of the written explanation on occasion. Again, a few paragraphs would suffice for a good answer.

Part (b) was generally answered well. State machines seem to be an area of relative strength for candidates. Areas where candidates lost marks included (in descending order of seriousness):

- Sometimes flow diagrams were given instead of UML state diagrams. Note: examiners can only give credit for the tasks that we ask you to perform in the question.
- Important states missing or poorly named.
- Flow of control not matching that in scenario.
- Missing operations on transitions.
- Errors in notation.

Question 4

This question aimed to examine candidates' understanding of sorting and heaps.

Part (a) was tackled well by most candidates. Some candidates showed the operation of a different sorting algorithm, others lost marks by missing key stages. A number of answers lost marks by unclear presentation that suggested that some did not understand the central ideas behind the algorithm. Good answers comprised of a full walk-through of the sort, plus a paragraph of explanation.

Candidates gave variable answers to (b), often confusing best and average case performance and not always explaining why the manner in that the list is unsorted can affect different sorting algorithms.

Part (c) was answered consistently well by candidates. There was occasional confusion about heap properties.

Part (d) was tackled well by most candidates. The main way of losing marks was not showing their workings clearly enough.

Question 5

This question aimed to examine candidates' understanding and application of linked lists and graphs.

Part (a) was answered very well. Candidates were generally able to explain clearly what a linked list was.

Part (b) and (c) were generally answered well, though some candidates gave confused answers.

Part (d) was tackled well by most candidates. The main way of losing marks was not showing working clearly.

Part (e) was answered consistently well by candidates. Again, marks were lost by not showing stages clearly.

Question 6

This question aimed to examine candidates' higher level understanding of hashing.

Part (a) was answered very well. Candidates were able to explain why hashing is useful.

Part (b) was tackled well by most candidates. A minority did not understand modulo arithm— candidates need to ensure they can do this. Other ways of

losing marks were not showing working clearly, and trying to resolve collisions when this was not asked for (though usually this did not result in a loss of marks unless we could not see the unresolved state).

Candidates gave good answers to part (c) and often were able to give a reason why collision is an issue.

Part (d) was answered consistently well by candidates.

Part (e) was answered surprisingly poorly by candidates, given it was rather straightforward. The usual mistake was to confuse a retrieval/search query (which was asked for) with an attempt to write the value in the hash table.