Examiners' commentary 2018–2019

CO2226 Software engineering, algorithm design and analysis – Zone A

General remarks

The examination was set generally as a mixture of questions that test candidates' basic understanding of the material, and questions that require candidates 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 is 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 plus 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 an answer, this 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 (by crossing it out). Examiners are directed to mark only the first two answers in each part. There were a few instances of this.

All questions require a mixture of bookwork (e.g. explaining terms, giving definitions) and application (e.g. calculations, building UML models) with the majority being application oriented. So candidates should work on practising 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 on 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.

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 gain no marks. If we see working, we can see what the candidate understands and they may gain partial marks.

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

Comments on specific questions

Question 1

This started with a UML-focused question where candidates were asked about the audience and scope of a UML activity diagram. This was followed by the usual 'generate a software engineering model from a scenario' task in the domain of a greetings' cards website (in this case, UML class diagrams).

a. Part (a) as noted above was bookwork. Candidates were generally able to explain what the diagram is (even though this was not asked for). It was when candidates were asked to say who it was for and what it captured, that the answers sometimes became vague and rather general. Some candidates unfortunately also failed to provide an example.

   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.

b. In part (b), the quality of class diagrams was variable but most did a good job and significantly better than in previous years. In general, candidates were able to capture the majority of the scenario somehow. There were a number of areas where candidates lost marks (in line with previous years).

   - 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.

Hint: the scenario is designed to give candidates a chance to show they can use all of the above. Candidates should avail themselves of this opportunity fully.

Question 2

This question started with a bookwork question on the use of an association class in a class diagram, followed by a scenario task (activity diagram, extending the scenario from 1(b).

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

b. Part (b) was generally answered well. Overall candidates were able to put most or all of the scenario on the diagram. But as in previous years, 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.

Hint: the scenario is designed to give candidates a chance to show they can use all of the above. Candidates should avail themselves of this opportunity fully.

Question 3

This question started with a bookwork question on the concepts of requirements churn, and then the development of a use case diagram based on the same scenario as the previous two questions.

a. Part (a) was generally not answered well, with many answers being vague. Examples were given in the stronger answers and sometimes lifted the quality of the written explanation. Again, a few paragraphs would suffice for a good answer.

b. Part (b) was generally answered well. Overall candidates were able to put most or all of the use cases on the diagram. In general, candidates were able to capture much of the scenario. That said, as in previous years common candidate errors included:

   - A few candidates gave use cases (as text) rather than use case diagrams. It is hard for examiners to give credit if the question is not answered as stated, so take time to read the question carefully.
   - Some candidates produced diagrams with missing actors (hint: sometimes time can be modelled as an actor).
   - More commonly candidates failed to fully illustrate, include, extend, and generalise relationships between the use cases, despite these being explicitly requested in the question.

Hint: the scenario is designed to give candidates a chance to show they can use all of the above. Candidates should avail themselves of this opportunity fully.

Question 4

This question aimed to examine candidates' understanding of ADTs and string matching.

a. Part (a) was not tackled well by many candidates. Many answers were confused about what an ADT is, and though they could usually describe the string data type they failed to provide operations. Candidates need to ensure that they understand this vitally important concept and why it differs from the concept of data structure (which it was often confused with).

b. Candidates gave variable answers to (b) depending on whether they recalled the bookwork accurately or not.

c. Part (c) was answered consistently well by candidates. Thankfully most candidates provided clear working for each stage.

d. Part (d) was tackled well by most candidates. The main way of losing marks was being vague about what the improvements were (rather than just naming them).

Question 5

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

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

b. Part (b) was tackled well by most candidates. A minority did not understand modulo arithmetic – candidates need to ensure they can do this. Other ways of potentially losing marks were not showing working clearly, and trying to resolve collisions when this was not asked for (though usually candidates did not in fact lose marks unless we could not see the unresolved state).

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

d. Part (d) was answered consistently well by candidates. A few candidates confused terminology and described/used a form of hashing that was different than what was asked for in (c) and (d).

e. Part (e) was sometimes answered poorly, given it was rather straightforward. The most common mistake was to confuse a retrieval/

search query (which was asked for) with an attempt to write the value in the hash table.

Question 6

This question aimed to examine candidates' understanding and application of graph representations and traversal.

a. Part (a) was answered very well, though with some confusion. Effective examples were usually given.

b. Part (b) was generally answered well, though some candidates gave confused answers to (or failed to address) incidence matrices. The topic of space-efficiency was generally answered well.

c. Same as for part (b) above.

d. Part (d) was tackled well by most candidates. The main way of losing marks was unclear or partial pseudo-code.

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