COMP61511 Exam performance feedback 2014

Question text in plain font, original marking scheme in bold, additional comments in bold italic.

On the scripts: / = 1 mark, X - wrong, ? = dubious or unclear point; judgement call whether to give the mark (usually not). Integer marks only except where noted below.

Overall I was pleased with the standard of the answers. There were a number of very good answers, along with some ordinary ones, but with very few really low marks.

Question 1

This question is COMPULSORY

a). State two ways in which Software Engineering is different from Physical Engineering disciplines such as bridge-building. (2 marks)

Requirements change is a much bigger issue [1]. Other mark for any other reasonable point, e.g. SW systems are much more complex dynamic behaviour.

No marks for saying that physical engineering uses physical objects while software engineering uses software ones!

b). In an agile process, users and stakeholders are expected to review partial versions of the system at the end of each iteration, and give feedback. State two other things they are expected to do. (2 marks).

Marks for any two reasonable points, e.g. they are expected to 1. Come up with requirements in the first place, 2. Participate in the production and review of user documentation, training etc.

A number of answers were about stakeholder feedback on the system, which was too close to exclusion in the question to be worth marks.

c). State a simple test for determining whether a candidate use case is of appropriate size, giving one example of use case which passes, the test and one which fails. You only need to state the names of the tests, not any details. (2 marks)

Boss test. If the end-users boss comes up and asks them what they’re doing, they ought to be able to give a sensible answer. [1] E.g. a navigation system - not “Turn left 10 degrees” (too small) or “Navigate ocean” (too big) but “Avoid storm”.
I didn’t give the mark for just the words “Boss test” as I needed to see whether people actually knew what it was. A number of answers had UCs at too fine a level of granularity.

d). You have drawn a domain model and as a result have questions for your customer. State two different ways you could go about this, depending on your customer’s background. (2 marks)

None-technical customer - write down a set of questions that arise from the model, and take that, not the diagram. [1] Customer with a technical background - take the diagram, with points of interest marked. [1]

No marks for general stuff like “respect the users expertise”

e). Give two examples of Pure Fabrications (other than UI classes) which would probably be useful in an aircraft design application. Hint: such an application would need to deal with a large number of parts, many of which would be similar to each other. (2 marks)

The hint suggests a PartFactory. [1] The other mark for anything reasonable, e.g. a database connector for a supplier parts database.

A number of answers suggested PFs which could apply to many different applications, not this one in particular. The other main error was saying that an abstract class representing similar parts (e.g. EnginePart) was a PF - such classes are valid domain concepts here.

f). Briefly explain what a statechart is, and how one could be used in the application of part e). (2 marks)

A statechart is a state transition diagram with nested states. [1] It could be used to model aircraft systems (such as pressurisation) as they go through different places of flight. (or anything non-trivial enough to require nested states) [1]

For the first mark, answers needed to explicitly say nested states - otherwise it’s like any other state diagram.

g). There are many factors which influence the bug density of an application. The amount of testing done is one; the nature of the application (e.g. whether it is reactive/concurrent) is another. State two other factors which are likely to increase bug density. (2 marks)

The converse of good practices, e.g. inexperienced developers leading to poor designs, single points of failure in the team (as opposed to pair programming) etc. Marks for two distinct valid points.

A number of answers talked about bug density in general, not giving specific factors which would increase it.
A contract violation is a bug [1]. A DbC tool will insert code to check for contract violations, hence giving you some testing for free. [1]

A number of answers showed understanding of DbC, but didn’t answer the specific question.

i). State two properties which dependencies between layers should have if possible in a layered application (2 marks)

They should be unidirectional [1]. In particular the business logic layer should not depend on anything else (although everything else will depend on it.) [1]

A lot of answers to this were rather vague. Yes low coupling is good, but I was looking for something more specific.

j). Briefly explain with an example the problem which Aspect Oriented Programming is designed to solve. (2 marks)

The problem is cross cutting concerns - ones which cross-cuts the natural class structure and make classes uncohesive. [1]. E.g. logging - without AOP we need to have code that says “do stuff, log that we’ve done it” all over the place (or any sensible example) [1] (An example from a specific application is good, but not necessary.)

Answers to this were generally good.

Question 2

This and question 3 were the most popular of the optional questions, although generally people who did questions 4 and 5 got better marks.

a). State four specific agile practices which you followed in your team project. (4 marks)

e.g.: working in time boxed iterations (imposed by the assessment schedule), frequent meetings with the “customer”, working in pairs, and responding to change in requirements by changing the plan. Note that I’m asking for specific practices, no marks for reciting the Agile Manifesto.

A number of people cited distinctly un-agile practices like having virtual meetings!

b) State two specific agile practices which you did not follow in your team project. (2 marks)
E.g. using user stories (as opposed to use cases), retrospectives.

A number of people correctly pointed out that they were required by the marking scheme to produce detailed and pretty models, which is un-agile.

c). Consider the statement: “In writing a use case, it’s important not to be too specific”.
To what extent is this accurate? (2 marks)

It’s important to abstract away for UI details - [1] but in general the statement is not accurate as we need to state specifically how the user will use the system.[1]

There were some good answers to this but others were vague and woffly.

d). You have been hired by the University of Mancunia to implement a University-wide timetabling system, to go on line for the start of the next academic year, in 9 months time. Currently, timetabling is done in various different ways in different schools, and over 200 admin staff and academics are involved. Under the new system everybody will use the same software and the timetable information will be stored in a central database which will form the “single point of truth” for all timetable information in the University. The software will be based on an existing system, which has only been used by two schools so far,

What will be the main risks associated with this development, as perceived by the following groups of stakeholders?:

Here there was a problem, as many people answers in terms of concerns rather than risks, and interpreting the question strictly, I could have given 0 to a lot of valid points. I therefore gave half a mark for points which implied risks, and marked the whole of part d out of 6, rounding up.

i). Students (1 mark)

Obviously that their timetables might be wrong. Or that they might not be able to access it at all.

ii). IT support staff (1 marks)

That the system may require extra work, e.g. having to manage a new, dedicated server. That they might get blamed if something goes wrong.

iii). University management (1 mark)

That there will be damage to the reputation of the University if it goes wrong (e.g. a story in a local newspaper, as happened at Salford, or worse).
iv). Admin staff responsible for timetabling (hint: consider both short-term and long-term risks from their perspective).

That they will not understand the new system well enough, and hence screw up; in the short term that the system will be more difficult to use than what they’re used to, and hence increase workload. In the long term, that the system will be more efficient than previous methods, and therefore require less admin staff to run it.

Many people interpreted “long term” to mean in 9 months time, which made for some strange answers. Generally I gave marks for reasonable risks/concerns, not just those cited above.

e). Of the four groups of stakeholders mentioned above, briefly state for each whether it would be important to have a representative interacting frequently with the development team. (4 marks).

Students - no; the best way to address their concern is to ensure the admin staff can use the system effectively.
Uni management - ditto
Admin staff - obviously yes. As well as getting feedback from them it will be essential to ensure that they have adequate documentation and training.
IT support - possibly, depending on how significant the IT infrastructure changes will be.

I gave marks for widely varying answers for this IT support, depending on what assumptions had been made about their role.

f). What would be an ideal system test for this system, and why? (2 marks)

To input the current timetable into the system. That way, we can discover any scalability issues, as well as training etc. before the system goes live.

Nobody came up with exactly this, but I gave marks for sensible suggestions for system tests.

Question 3

a). Explain what ceremony is, and give three specific examples of how the effort spent on it can be reduced in an agile project compared to a traditional one. (4 marks)

Ceremony = anything which isn’t part of the end product. (That’s not the same as everything except the code). Examples: don’t produce fully dressed UCs unless there’s a good reason; don’t produce documents that largely replicate existing ones (e.g. a UP vision document and a funding document) draw diagrams on a
whiteboard and take a photo rather than using a tool; don’t draw over-detailed diagrams.

b). State four different ways in which a domain model can be useful. (4 marks)

It gives us an overview of important domain concepts and their relationships; inspecting it can often generate questions to go back to the stakeholder with; it often reveals information about process; it forms a staring point for design. (Note although the first two parts are bookwork, I expect most students will struggle to come up with enough distinct points)

A number of people said domain class diagrams are used to communicate with users. No marks for this unless it was qualified in some way, as non-technical users are not going to understand the diagram.

c). As part of designing a timetabling system for a University, you are interviewing an administrator, focussing on what is considered important from their point of view about rooms. She tells you:

“For every room we have its location, and how many seats it has. Otherwise, the information we hold is different for each type of room. For a lecture theatre, we record what kind of equipment it has (e.g. blackboard, data projector). For a computer lab we record how many computers it has, and for a tutorial room, since such a room may be a staff office, we record what member of staff, if any, occupies that office. Finally, we have a category of “special” rooms, e.g. biology labs, for which we just have a note of what they are, as we’re not involved in timetabling them.

i). List the important domain classes implied by this description. (4 marks)

Room, LectureRoom, ComputerLab, TutorialRoom, SpecialRoom, Location, Equipment, StaffMember is my optimal list and for full marks all these should be present. BlackBoard and DataProjector are acceptable, although these are just examples of equipment, and it’s not clear that these need to correspond to classes.

ii) Draw a domain class diagram which shows the relationships between these classes
You may assume that a room serves only one function at any one time, and if the function of a room does change, it is treated as a completely new room.
Note: if there is no enough information in the above description to model a concept in detail, do not attempt to do so. (5 marks)
First, the middle sentence of the question tells us that this is not an instance of the Roles problem, so we can subclass directly from Room (although an ultra-cautious solution that treats the functions of rooms as roles is acceptable).

I’ve modelled Location, StaffMember and Equipment as classes as they are likely to be compound things, but not attempted to model them in detail. I’ve also assumed that the number of computers in a computer lab is not necessarily the same as the number of seats - making the opposite assumption is also ok. Full marks for the above or minor variants (in particular modelling different kinds of equipment. 4/5 for the above but without the multiplicies.

Marks deducted for:
- Important concepts missing
- Additional stuff/clutter not justified by the description
- Abuse of UML notation.

There were many good solutions to this. The most common error was not recognising that a location is a first-class concept and ought to be a class. I was severe on diagrams which included types, especially inappropriate ones such as location: String. (Location information is important as you don’t want to run the students and staff around the campus any more than necessary)
d). Give three follow-up questions you would need to ask the administrator in order to turn the domain model into a concrete design. (3 marks)

   It makes sense to focus on the under-specified parts, e.g.
   “How do you specify the location of a room within the University?”
   “How much information do you need about the equipment in a lecture room?” (If it’s just a checklist we don’t need to model pieces of equipment explicitly)
   “What details do you need about members of staff who occupy tutorial rooms?” (if it’s just a name, we don’t need a separate class, if more than that we do)
   Marks for any three sensible questions, given in language the administrator will understand. Could also ask whether number of seats is the same as number of computers in a computer lab.

   Many questions were too vague and general. In particular the “what is your role” question is a good one to ask at the beginning of an interview, not at this stage.

Question 4

   This question was relatively unpopular, but answered well by most of the people who did it. In particular most people got full marks for parts a) and b)

a). Throughout the course, the importance of communication in Software Engineering was stressed. Give three different examples of techniques for communication between developers and stakeholders taught in the course. (3 marks)

e.g.
   Asking questions appropriate to the stakeholders expertise
   Domain modelling, particularly asking questions arising from an initial domain model
   Giving demos of partial systems and getting feedback.

b). Give three different examples of techniques for communication among developers taught in the course. (3 marks)

e.g.
   Collaborative design at a whiteboard
   Discussing design choices using the names of GRASP principles as shorthand
   Agile practices, e.g. stand-up meetings.

c). Briefly explain the GRASP principles of Polymorphism and Protected Variations, using an example not covered in the course or elsewhere in this exam. (4 marks)

   Polymorphism = use inheritance to improve cohesion and coupling. PV = design the code so that when something changes, impact is minimised. Marks for any valid use
of inheritance where we can add subclasses, or change the implementation of a
subclass, without changing anything else.

d). Explain, giving specific examples, how your MELT design conformed to six different
GRASP principles. You may count the two kinds of coupling as different principles
provided you distinguish clearly between them. If your design did not conform to six,
state how it should have been improved to do so.

High cohesion - e.g. there should be clearly defined core classes like Section
Question, etc, clearly separated from the interface code
Low internal coupling - e.g. within the UI code there should be a separate class
dealing with each separate part of the UI, with low coupling between them.
Low external coupling - e.g. there ought to be clear model-view separation, with or
without a Controller.
Polymorphism - there is relatively little scope for this in Melt, but they should have
a Question class and subclasses for each individual question type, and hence
Protected Variations (as in part C).
Information expert (needs a specific example of operation-data co-location).
(Could also have Indirection, Creator, Controller or Pure Fabrication).

The main reason people lost marks here was not clearly relating the principles to the
specifics of their MELT design. There were also a number of confessions, which was
good.

e). Explain the relationship between the GRASP Creator principle and the use of a
Factory. (2 marks)

They are alternatives. (Common error is to think Factory is an application of
Creator). Creator gives a set of suggestions for places to create objects within
existing classes. A Factory is a separate Pure Fabrication used solely for this
purpose.

Unlike on my third year course, a number of people got this right. Well done to those
who did.

f). Suggest how a factory could be used in MELT (other than the example given in the
lectures). (2 marks).

The example given in the lectures was generating questions for adaptive testing.
Other uses are non-obvious (e.g. creating adaptors for different question formats).
Marks for plausible suggestions.
Question 5

This was the least popular question, but most of the people who answered it knew what they were doing. A couple of people couldn’t answer part d, making it a strange choice of question.

a). Briefly state the difference between integration testing and system testing, according to the terminology used in the lectures, and give three examples of things which need to be done in system testing but not necessarily in integration testing. (4 marks)

The terminology I use is integration testing is testing the system, or a subsystem, in the development environment, while system testing is testing in the deployment environment.

(This is one of many different uses of the terms in the books. I use the terminology this way as I think it’s important to make this distinction, and also between system testing and acceptance testing, which may be considered a subset.)

We need, if possible, to test with a range of real users (not just out pet customer reps) who will use the system in different ways. We need to ensure that the documentation is suitable for those users, and identify additional documentation or training needs, as well as modifications which may need to be made to the software itself.

Answers to this were mostly good.

b). Explain the term “bug density” as used in the lectures, and state how bug density is related to the total size of the code, and why. (3 marks)

Bug density is the number of bugs per unit lines of code (e.g. 1 per 100). [1] On average, it goes up as the number of lines of code increase. [1] Reason along the lines of: This is because it’s harder to come up with a clean simple design, there are more interactions between parts of the system, and hence a higher proportion of the more difficult kinds of testing [1]

A number of answers amounted to saying the number of bugs increases with size (which is obvious!) as opposed to bug density.

c). Estimate how many bugs there are likely to be left in your MELT implementation. Explain your reasoning, taking into account the estimates of bug density given in the lectures, and the factors relating to bug density in your project. (5 marks)

The estimate given in the lectures is 1-10 bugs per 100 lines of code which just compiles, of which 90% will be found by routine debugging (which they probably did) and another 90% by routine testing (which they may have done -and another 90% by really thorough testing, which they are unlikely to have done). Hence if they
have 3,000 lines of code (a typical number) they will have started with 30-300 and
got that down, in theory to 0.3-3. In practice due to inexperience, lack of time etc.
they’re unlikely to have achieved that, and something around 10 is a more
reasonable estimate. Answers to this kind of question tend to have very high
numbers (if it has 100 bugs, nothing will work!) Full marks only for answers which
give a reasonable estimate based on a reasonable rationale.

The estimates tended to be much smaller and more realistic this time round.

d). For each of the following, explain its purpose in JUnit testing, and outline how it
expressed in JUnit code (exactly correct syntax is not required).

i). An assertion. (2 marks)

The basic component of a JUnit test, which will throw an AssertionError if it fails,
e.g.
AssertTrue(x == 0)

ii). A fixture (2 marks)

Some code which is run before a test, or set of tests, is run, to set things up.
Expressed with annotations @Before or @BeforeClass, e.g.

@Before
public void setup() …

Note: due to an error on the slides, answers which include @After in the fixture will
be accepted.

ii) A test suite (2 marks)

A set of tests to be run together. Expressed using reflection, e.g

@Runwith (Suite.class)
@TestClass(test1.class, test2.class…)
// Dummy class to give the assertions something to attach to.

This part was answered correctly by most of the people who attempted it.

e). State two limitations of JUnit testing, giving examples from using JUnit on your
MELT project.

There is no good way to test UI code with JUnit. Classes depend on other classes, so
“Unit” testing is not a simple as it sounds.

And most people got this right.