General comments: while question 1 was intended to assess breadth of knowledge, and was largely answered well, the other questions were designed to be quite tricky in places, and to have a slightly lower average and higher standard deviation than in the guidelines, which is what happened.

A few people (although less than I feared) showed clear signs of having difficulty with the exam due to English problems. Be warned: writing a dissertation is MUCH more demanding of your English skills than answering an exam.

There was an obsession with Controllers (in the MVC sense) which I didn’t understand because I didn’t make a big deal of them in the course. This had a particularly strange effect on Q2.

Question 1
This question is COMPULSORY.

a). Briefly explain the relationship between the Unified Process and Agile development. (2 marks)

We can apply the UP in an agile way, where all Ceremony (artefacts which are not part of the final product) is optional. [1]
For the second mark, one of: those artefacts we do produce can be low cost, e.g. a photo of a diagram on a whiteboard; we can apply many agile practices within the UP; the UP is similar to Agile in its structure, having timeboxed iterations.

b). One of the things which went wrong with the Waterfall method of development was that it doesn’t cope well with requirements change. State 2 others. (2 marks)

Two of: specialisation within teams can lead to poor communication and difficulty managing resource; fear of the cost of fixing things downstream leads to Analysis Paralysis; there is no customer feedback until near the end; if the budget runs out nothing useful is produced; or other valid points.

c) In almost any system the following is a possible use case:

Authenticate user
1. The user types their username into the box marked “username”
2. The user navigates to the box marked “password” by hitting the Tab key
3. The user types their password into that box
4. The user hits the Enter key to confirm
5. The system allows access if the username and password are correct, otherwise it displays an error message and presents the boxes again.

Briefly state two reasons why this is unlikely to be an appropriate use case. (2 marks)

It is at too low a level of granularity - doesn’t pass the Boss Test. [1]. It fails to abstract away from UI details.[1]

There were a lot of strange answers to this, in particular many people didn’t even mention the Boss Test, and some of those who did didn’t actually seem to know what it means. UC granularity was something I put a lot of stress on so there’s really no excuse.

Some people said the name of the UC is inappropriate. While just “log on” would be simpler, it would be less accurate. For example, in order to perform the UC “View MSc project allocation” (boss test - “I’m finding out who my MSc project students are and what projects they’re doing”) I need to authenticate myself even though I’m already logged on. Some people claimed that terms like “tab key” were technical language. Some people criticised step 5 as it doesn’t explicitly involve the user, but this is normal - the system is responding to the actions of the user in steps 1-4.

d). What are the two main reasons for doing Domain Modelling? (2 marks)

To facilitate communication between developers and stakeholders [1]. To provide a starting point for design, as domain classes can “inspire” design classes. [1]

A lot of people missed the second point completely, instead elaborating on the first.

e). Give two examples of Pure Fabrications (other than UI classes) which would probably be required in a timetabling application. (2 marks)

Such an application will need a database of rooms etc., so a database connector will be required. [1] There will need to be some sort of entity containing an algorithm to generate a timetable. [1]. Other answers accepted provided they are specifically related to this application. Note: this is quite tricky as there are a lot of domain classes here which would likely correspond to design classes.
While most people knew what a PF was (although note that an abstract class is not necessarily a PF, it may correspond to a domain concept) a lot of answers were generic and not specific to a timetabling app.

f). There are many factors which influence the bug density of an application. The amount of testing done is obviously one; the nature of the application itself is another. State two other factors which are likely to decrease bug density.

(2 marks)

Two of: choosing a relatively safe programming language such as Java; the nature of testing either having a separate test team or doing TDD; programming in pairs or other valid points.

Mostly well answered, some answers cited factors which affect bug density rather than specifically decrease it.

g). Briefly explain what a statechart is, and how one could be used in the MELT application.

(2 marks)

A statechart is a finite state diagram with nested states. [1] Second mark for anything sensible, e.g modelling the states of a student during the process of taking a test (waiting, ready to start, working [with substates] timed out etc.

Again mostly well answered but some people confused states with classes!

h). State two properties which the partitions within a layer should have in a well-designed application.

They should all contain code to do similar things, corresponding to the layer, e.g. UI code, data storage code etc. [1] They should have low external coupling, i.e. they should have few dependencies on partitions in other layers and those dependencies that do exist should ideally be simple and unidirectional. [1]

A number of people just said high cohesion and low coupling, which is true an any classes. Several variations on the answer above were accepted, so long as they specifically addressed partitions and layers.

i). How is Design by Contract related to Java exception handling?

(2 marks).

A contract violation is a bug [1]. It should therefore lead to an unchecked exception (a RuntimeException or an Error [1]).

A lot of answers had the right sort of words in but weren’t accurate enough for full marks.
j). Apart from Aspects themselves, what are the two key features of the AspectJ programming language? You do not need to show any AspectJ syntax. 

(2 marks)

**Pointcuts** are well defined points within the execution of a program. [1] *Advice* is code to be executed when a pointcut is reached. [1] (Note for this purpose the terms “join point” and “pointcut” can be considered synonymous).

**Most people knew this.**

Question 2

Process and requirements

a). In what ways was the process you followed in the team project similar to the Agile UP. (4 marks)

Development was done in timeboxed iterations (at least timeboxed by the assessments!). [1] The artefacts produced were UP artefacts such as use cases and domain models.[1]. There was frequent interaction with the customer, including weekly demos. [1]. An agile approach to teamworking was used, e.g working in pairs. [1]

**Most answers to this were good. Although few mentioned the second point above, this was compensated for by mention of other agile practices the team used.**

b). In what ways was the process you followed in the team project different from the Agile UP. (2 marks)

There was no direct equivalent of the phases of the UP (Inception, Elaboration etc.). [1] Artefacts were in many cases not optional (they were required by the marking scheme). [1].

**Many answers to this were in the form of confessions, which I really liked and thought much more interesting than the first point above.**

c). State two important rules to follow when eliciting requirements from stakeholders with only basic IT skills. NB: your answer should be specific to that kind of stakeholder, not one that could apply to any stakeholders.

Avoid technical jargon; learn and use their terminology instead. [1] When doing domain modelling follow-up, prepare a list of questions from the diagram and take that rather than the diagram itself. [1] Also acceptable is respect and make use of their particular area of expertise.
Mostly well answered, although a number of people advocated showing them the diagram which is not recommended for this class of stakeholder.

d). You have been hired by the Highwayman Bus Company to develop a Second Generation Bus Tracking and management System. Currently the Company tracks all its buses by GPS, but the existing system does little more than tell managers where their buses are at any one time, using a crude visual display.

The new system will make may improvements, including making real-time information available to passengers, greatly improving the managers' interface, and comparing the performance of individual drivers. The long-term aim is to maximize revenue by optimizing the usage of buses and drivers. The system will be operated by Controllers, who are responsible for ensuring that services run reliably on a day-to-day basis.

For the following three groups of stakeholders in the system, suggest one short-term, and one long-term, concern they will have about the system. Hint: think about the consequences of the sentence "The long term aim is.." in the preceding description.

In all cases below, sensible alternatives are fine so long as they answer the question as stated.

i). Passengers (2 marks)

**Short term:** that the real-time information provided is accurate - otherwise it is worse than useless

**Long term:** that the "optimization" will not adversely affect services.

ii) Drivers (2 marks)

They will not be happy!

**Short term:** that performance monitoring will not impose undue pressure, e.g. to drive too fast.

**Longer term:** that "optimization" could have all sorts of unwanted consequences, such as changes to working hours and practices, and even redundancies.

iii). Company management. (2 marks)

**Short term:** that the system will be delivered on time, on budget, and without undue disruption.
Long term: that they are indeed able to optimize services and hence the system will be of net financial benefit

A mix of answers here, some very good. Note that a concern is not just a functional requirement. A number of answers were somewhat optimistic about the drivers’ attitude to the system! A number of answers made different assumptions to mine (in particular that drivers would interact with the system directly in some way) and plausible concerns based on those assumptions were accepted.

f) Discuss whether it would be sensible to have a single stakeholder representative work with the project development team. (4 marks)

An ideal answer is along the lines of:

If we do have such a representative that person will need to be a Controller, or at least someone with experience of that role, as they are the primary users. To check whether it is ok, look at the other stakeholders. For passengers, any real-time information would be better than none (and there are many examples around of how to display such information) so talking to them is not high priority. The drivers’ concerns are not our problem and they don't use the system directly. Management are buying the system and have specific aspirations for it, but a Controller, who must themselves be fairly senior, can represent these. Hence a single representative is appropriate provided they have the right background.

Marks for any reasonable, focussed, discussion.

Many people didn’t notice that the Controller (a person not a software arefact!!!) was a user at all, never mind the primary one. None of the answers looked anything like the above. A number of people argued (usually on the assumption that drivers were directly involved) that there was no one such person and I accepted such answers provided they did include the Controller.

g) As the project progresses, you find that the controllers’ user interface which is of course complex, continually causes problems. - each iteration produces some improvements but also reveals further issues. What would you do to try to get a grip on the situation? (2 marks)

Get together with the customer. [1] Second mark for anything sensible and specific, e.g. Consider what you could do which might improve the situation - e.g. a major refactoring exercise, or agreeing a change in specification of the interface.

Here, a number of people were still thinking “controller” as in MVC, leading to some very confused answers. Of those who understood the question, a number failed to mention getting together with the customer, and/or suggested general ideas not specific to the situation.
Question 3

Domain modelling

You have been hired to implement software for a Safety and Security system for a large building. Currently the building has many safety and security devices, but very primitive facilities for monitoring them - essentially one large panel of lights for security and another, in a completely different place, for fire safety.

a). You are interviewing the Head of Safety and Security for the building to find out more about what's required. Suggest five questions it would be sensible to ask to start off with. (5 marks).

1. Personally (this was the subject of some discussion during the course) unless I'm absolutely sure of somebody's role/job description I ask them about that first. In this case is he just a manager or a hands-on guy who knows a lot of technical details? People in this sort of job are usually of the latter kind, which means we can ask technical questions. Let's assume this (students who made different assumptions, will have different questions), then possible follow-ups are:

2. Roughly how many devices, and how many different types do you have? (Important to get a sense of scale)

3. Will the new system use the existing devices, or is the plan to introduce extra or different ones?

4. Do you want to monitor security and fire safety together, or are these regarded as distinct roles?

5. Could you give me a general overview of how you would like the new system to operate? (this is of course the important one, but by asking a few "scoping" questions before we're more likely to understand the answer and not be talking at cross-purposes).

It was of course possible to reverse-engineer questions from the answers below, and nothing in the question prevents that. However, such answers got a maximum of 3/5, because questions so generated are likely to be too specific, and too similar to each other, “to start off with”.

b). He gives you the following information:

"We have 30 floors with around 30 devices on each so around 1000 total. The majority of these are smoke detectors but there are door alarms and a number of other kinds, maybe 10 in total. The fire safety devices are of high standard - they have to be - but we would like to improve security, in particular we would like to make more use of CCTV."
We do want to monitor both fire safety and security together, but we want that monitoring to be distributed over the building in a flexible way. What we want is a monitoring station on every third floor, but only to have them all manned at busy periods. So at other times (e.g. overnight or at weekends or if we're just short of security staff) we would use less monitoring stations. There are also times when we may want more than one monitoring station to monitor a particular group of devices - if we're making a decision about whether to evacuate the building, for instance. Hence the challenge for you guys is to enable the monitoring stations to "see" different groups of devices at particular times."

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

They should have a list very similar to the following:

Device, DeviceGroup, FireSafetyDevice, SecurityDevice, SmokeDetector, CCTV, MonitoringStation, Floor

Most people missed a number of these, in particular nobody noticed the need for a DeviceGroup concept. A number of people included human actors, which is fine provided the other concepts are also present.

ii) Draw a domain class diagram which shows the relationships between these classes. (5 marks)

The diagram should have an inheritance hierarchy of devices, the notion of a device group containing devices, and a many-many relationship between devices (or device groups) and monitoring stations.

The answers to this reflected the limitations noted for the previous part. At least most people used the notation correctly.

c). You identify that the project has three main aspects: the user interface for the monitoring stations; the protocol which implements the many-many relationship between devices and monitoring stations, and management functions which summarise the activity which takes place over a given time period. In what (if any) order would you tackle these aspects, and why? (3 marks)

The rule is to tackle the high-risk aspects of the project first. The management summary stuff can be left until last, but it would be sensible to have logging from the beginning so that the raw data is there (to be used for troubleshooting purposes initially). Although the user interface is important, the ability to connect devices to monitoring stations in the required manner is critical (especially as it may involve physical wiring and low-level device management and it needs to be done without disrupting the existing system). Doing this for real would be darned tricky
Most people had the protocol first, but (partly due to a different interpretation of “management functions”) had the other two the other way round. I allowed this iff there was a plausible argument - generally you want to build UIs early to facilitate customer feedback.

d). A device being triggered can be considered as an event, just like a button being pressed in a Java UI. This suggests a way of providing the flexibility required in associating devices with monitoring stations - can you see what it is?

An object representing a device could hold a list of the monitoring stations to which it is associated (there should always be at least one). The set monitoring stations with which a device is associated is changed simply by adding or removing them from the list. When the device is activated, an object representing the event which occurred is sent to all the monitoring stations on the list.

i.e. the Observer design pattern. I’d only expect very strong students to figure this out

Controllers were extremely popular here! I gave one mark for a plausible suggestion of how one might be used, although it doesn’t directly address what’s required here.

Question 4

Design/GRASP

a) Explain with an example the relationship between the GRASP principles of High Cohesion and Low Coupling, and the practice of Refactoring. (4 marks)

The purpose of refactoring is to improve our design, and the GRASP principles give us guidelines on what constitutes a "good" design. Our initial design may show poor cohesion (classes doing more than one job) and unnecessary coupling. For example suppose we have a Roadvehicle class in a traffic simulation application, and it becomes uncohesive because of the different characteristics of different vehicles. We can refactor to make RoadVehicle abstract, with subclasses for different kinds of vehicles. The resulting classes will be more cohesive and the rest of the code is coupled only to the interface of the Roadvehicle class.

Many answers failed to establish a clear connection between the two things (some people didn’t appear to know what refactoring is; it was in the course but I need to emphasize it more next year). Others failed to give an example. A number of people said that having high cohesion and low coupling makes refactoring easier, which I allowed if expressed coherently.

b). Briefly explain what the two types of coupling are, and for each give a specific example of how you kept it low in your MELT implementation. (4 marks)
*Internal coupling* is between the classes which make up a subsystem. *External coupling* is between subsystems. An obvious example of the latter is model-view separation (another is separation of the business logic from the data storage). Their example of internal coupling will depend on the details of their design. For full marks the answer should be specific to their implementation, ideally naming particular classes, not generic.

**Most people got this at least roughly right.**

c). One function of MELT is to output results. Initially, you have a ResultGenerator class which generates results in the format required by the English Language Centre. Later, you discover a requirement to provide results to the University central administration, but in a different format, so you add code to the ResultGenerator class to do this. Then, you discover that different schools within the University also require results, and also have different formats. The ResultGenerator class is becoming large and uncohesive.

Draw a skeleton design class diagram which shows how **polymorphism** can be used to improve cohesion in this situation. You should assume that the results are always in a form of a spreadsheet, and that only the details of the information on the spreadsheet differ. (3 marks)

![Diagram](image)

Reasonable alternative names for the subclasses are fine. Note my non-standard notation for abstract, which I’ve commended to them. Showing the results generation method in the subclasses is acceptable, although I prefer to leave it out as it’s clutter, and is implied by its appearance as abstract in the superclass. Other
clutter will be penalized, as will incorrect use of notation (e.g. diamond rather than triangle.

Of those answers which looked plausible, the main problem was having a class hierarchy of result types, rather than result generators. The problem with this is that you have to put the generating code, which depends on result type, somewhere, and if you put it in the ResultGenerator class it remains uncohesive, while if you put it in the Result classes they are mis-named and the ResultGenerator class itself becomes redundant.

d) What is the effect of this use of polymorphism on coupling? You should consider both kinds of coupling mentioned in part c. (2 marks)

Internal coupling is minimal, each subclass depends only on the superclass, which depends on nothing. [1] External coupling is reduced as external classes have a simplified interface to work to. [1].

My original standard answer above actually makes the same mistake of several actual answers, of being generic rather than specific to this example!

e). How does your design support the principle of Protected Variations? (2 marks)

PV means we protect against variations which may happen in the future. [1] In this case, adding a new results generator just means adding a new subclass; no other code is affected. [1] Also, changing the details of one results generator does not affect anything else.

This was generally well answered.

f) Briefly explain what a **factory** is and how it could be used in the design given in your answer to part c (3 marks).

A factory is a Pure Fabrication whose sole purpose is to create objects. [1] In this case the objects to be created are instances of the results generator subclasses. [1] The appropriate subclass instance to create would be determined by e.g. a parameter being provided or via a config file. [1].

Answer to this depended on the assumptions made previously. Few people made three distinct points.

g). Now suppose that the assumption stated above, that the results are always in the form of a spreadsheet, does not hold true; results can be in many different forms, such as XML or relational database tables. Draw a skeleton design class diagram which shows you your design in part d can be enhanced to deal with this situation. (2 marks)
Anything which gets the right general idea (without abusing the notation etc.) gets the marks; in particular the generates > relationship can be omitted as it is implied by the operation. Having a Result subclass corresponding to each ResultGenerator is also ok.

_A lot of people here got 1 mark for something along the right general lines, depending on their answer to part c._

---

**Question 5**

**Testing**

a) What is the difference between _black box_ testing and _white box_ testing? Give an example of how each could be used in MELT. (3 marks)

_Black box testing is done from the interface of the code being tested, while white box makes use of knowledge of the code itself._ [1] Marks for any reasonable examples,
e.g. black box - behavioural testing of the UI, while box, testing of some internal logic which has caused problems.

a) What is the difference between integration testing and system testing? Give an example of how each could be used in MELT. (3 marks)

Integration testing is testing of a system or subsystem in the development environment. System testing is testing of a system (and associated documentation, procedures etc.) in a deployment environment. [1]. E.g. integration testing - making sure model, view and controller are working together correct. System testing - having the “customer” try out the setter’s interface.

These first two parts were generally well answered.

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

The relevant estimates are: 1-10 bugs per hundred lines, of which 90% may be found by debugging (which they did do) of which 90% may be found by “routine” testing (which they probably only partially did). Factors affecting bug density include programmer experience and whether they practiced programming in pairs.

First they have to estimate how many lines of code they have. Let’s use 3,000 as an example. That’s 3-30 bugs after debugging. In theory that’s 0.3-3 after testing, but it’s unlikely that they did the amount of testing required to get that order of magnitude. If they have a clean design, tested really thoroughly, and programmed in pairs, there’s a chance their code could be bug free. Much more likely, given the relative inexperience of most members of most teams, and the strong time constraints on the amount of testing done, there will be several bugs remaining, where “several” could easily be 10 and might be as many as 30. A good answer will combine the estimates given in the lectures with the reality of their experience.

Answers to this were fascinating, and often involved confessions - clearly few people were confident of the effectiveness of their testing! Estimates of the size of MELT implementations varied from 900 lines to 20K. Estimates of the number of bugs were ever more variable - from one or two to 700! The vast majority were much higher than my suggestion above, often in three figures; since all teams had a decent amount of working code to demonstrate many of these estimates must have been well high - clearly people were scared to suggest that their code was anywhere close to being thoroughly tested.

I gave marks mostly on the basis of taking appropriate factors into account, rather than the final numbers. Most answers go reasonable marks, except for ones which failed to do basic things like estimate the number of lines of code.
d). Explain, using an example not given in the lectures, the key aspects of JUnit testing. (4 marks)

JUnit testing tests one unit (a class, or some people like to say a method) at a time, using test code separate from the class being tested. [1]. It is based on assertions which test whether particular properties are true; if they are not, and AssertionError results. [1].

The example given in the lectures is sorting. They will probably use an example from their MELT implementation. A possible example is putting a value into a Map. Initially it’s not there:

```
assertFalse(map.containsKey(key))
```

after we put it in
```
map.put(key, value)
```

it’s there and we can get it out:
```
assertTrue(map.containsKey(key))
assertTrue(map.get(key) == value)
```

2 marks for any sensible example.

*A lot of answers to this were poorly expressed. A lot of people fixated on fixtures and never actually mentioned assertions.*

e). State three ways in which JUnit testing is beneficial. (3 marks)

It gives you a level of confidence in the individual units before you try to integrate them. [1] The tests can be automated so its good for regression testing etc/ [1] By separating the tests from the code, it means if the code is refactored the tests remain valid. [1] (Also helps to clarify the interface of the code).

*Generally answered ok, although a number of people failed to make three distinct points, in particular automation of testing was rarely mentioned (presumably because that wasn’t done in the lab).*

f). State two disadvantages, or limitations, of JUnit testing. (2 marks)

It can’t be used for testing the behavioural aspects such as user interfaces. [1] Almost every class depends on other classes, so if an assertion fails it doesn’t necessarily mean that the unit being tested is at fault. [1]

*There were a number of rather complicated and unfocussed answers to this. The two simple and obvious points above were sufficient.*