Comments:

General comments: the exam turned out to be on the easy side, especially question 2, and hence was marked strictly, although the average is still on the high side. While most people did well, there were a small minority who did very badly indeed and had clearly not engaged with the course at all.

There were a significant number of marks lost due to failing to use the minimal subset of UML correctly, although probably not as many in previous years.

Question 1
This question is COMPULSORY

a) State four reasons why design patterns are important in object-oriented software development. (4 marks)

Provide tried and tested solutions to common software design problems
Provide a language which allow experiences developers to communicate efficiently
Provide a learning aid to inexperienced developers
Provide a framework for discussing design tradeoffs (and any other reasonable points)

This is bookwork and has been asked before and the above answer given in previous epf (indeed one person reproduced all five lines exactly!). Some people failed to come up with four distinct points but most answers got full marks.

b). The exam software has a tool which allows students to draw diagrams which consist of boxes joined by connectors. A box has content (inside it) and up to four labels around it. A connector always joins two boxes and has up to six labels attached to it. Since diagrams have a graph structure, we refer to them within the project as graphs. We can also take a complete graph and enclose it within a box, which is called a graph box.

Draw a design class diagram which captures the information above and makes use of the Composite design pattern to deal with graph boxes. You should assume that there is only one Box class and one Connector class, and the differences between particular boxes and particular connectors are represented by properties of these classes which you do not need to model. (6 marks)

The important part is the dual relationship between GraphBox and GraphItem which constitutes the Composite pattern. Using Graph directly instead of GraphItem is also fine, as are sensible variations on names. 4 marks for this part, 2 for the rest. As always abuse of the notation will be penalised (1 mark per kind of abuse) as well as inclusion of extraneous stuff.

This was done very poorly by many. Many answers did not show Composite as such, just some kind of inheritance, some not even that. Although there were a number of variants which were acceptable (in particular having Box as the top of the hierarchy), many of the solutions failed to follow the basic is-a-kind of rule for inheritance.

c). Since diagrams have a graph structure, we can compare them by graph matching which potentially gives us the ability to do some semi-automatic marking. This was in fact investigated by a PhD student and it turns out to be very complex. No single graph matching algorithm will do the job, we need several algorithms, each of which may be complex.

Briefly explain how the Strategy design pattern can be used to help manage the complexity involved. (3 marks)

Each algorithm is encapsulated within a class. [1] We have an abstract superclass and a subclass for each algorithm (strategy). [1] This gives us a simple, uniform interface for our algorithms and gives us a convenient way to switch between algorithms at runtime. [1]

Mostly this part was answered well.

d). Briefly explain how the Composite pattern can be used in conjunction with the Strategy pattern in this situation. Hint: this is a different use of Composite from that in part b. (3 marks)

We introduce the notion of a CompositeStrategy which collects together a number of strategies which may themselves be composite. [1] This allows us to build an arbitrary tree structure of strategy objects - graph matching algorithms [1] This allows us to apply an arbitrary combination of graph matching algorithms, both different ones and the same one repeatedly, as appropriate. [1]

Note: for full marks an answer must relate to graph matching, not a generic description of Composite strategies or the example in the book.

Some people failed to follow the hint, otherwise answers to this were generally ok.

f). State four different GRASP principles which are applied in both the Composite and Strategy patterns. (4 marks)

Four of:
Most obviously Polymorphism, both make use of inheritance
Indirection - code using the code in the subclasses uses the superclass interface
High cohesion - each class represents a single well-defined entity e.g. each Strategy class represents an algorithm.
Low coupling - code using e.g. a box or an algorithm is decoupled from the details of the concrete classes.
Protected variations, adding a new algorithm, of changing an existing one, doesn’t affect any other code; likewise for graph items, e.g. we could make Box a superclass and have different kinds of boxes, and external code would not be affected.

A problem arose in marking this question, because some people just listed the names of four GRASP principles which left me no way of knowing, if they were correct, whether they were just guesses (particular as a number of answers of this kind did not list a correct set. I gave these answers 2/4.

20 March 2013
A very common error was to list Pure Fabrication as one. Strategies are generally PFs, but Composites often arise from domain concepts (as is the case here where the domain concept “diagram” has simply been renamed “graph”).

Question 2

This question was too easy, in particular I should have asked for two specific Agile practices for each principle in part d. It was therefore marked strictly as I reasonably could.

Process and stakeholders

a). “The Unified Process requires the construction of many artefacts, many of which are Ceremony. Therefore the notion of the Agile UP is a contradiction in terms”. Give a concise criticism of the above statement, making sure you make two distinct, relevant points. (2 marks)

Firstly, the statement is incorrect, the UP does not *require* any of the ceremony artefacts (ones which aren't part of the final product: they are all optional. [1] Secondly, in the Agile UP we follow many agile practices such as working in timeboxed iterations and having constant contact with the stakeholders, and we can introduce others such as programming in pairs if we wish to. [1]

Many answers failed to make a second distinct point such as the second above, instead restating the first, sometimes in several ways. A few lost a mark for being far from concise.

b). Explain the role of users and other stakeholders in the UP (2 marks)

Stakeholders are consulted frequently throughout the project to ensure that the system being developed is appropriate to their needs. [1] In particular, staring early in the project, they see the state of the product at the end of each iteration and give feedback on it. [1]

This was mostly done well, although some people answered the very different question “What is the difference between users and stakeholders?”

c). List four distinct groups of users/stakeholders of the exam software other than students, and in each case state two important concerns that they will have (i.e. 8 different concerns in total). (8 marks)

Note: the four given below are probably the most important, but there are many others, particularly system administrators, but also potential students, parents (or other sponsors) of students, potential employers etc. Of course concerns other than the following are fine so long as they are important and relevant to the stakeholders concerned.

University management who (in the scenario) are paying for the software will want to ensure that it delivers real benefits to the University and represents value for money. They will also be concerned that the examination process is sound, so there are no bad reports in the newspapers.

Academic staff will want to be able to set and mark exams in a way that is appropriate to their courses, and not be overlay constrained by the software. They will also want setting and marking to be as easy as possible, and for training to be available as necessary.

e-learning technologists will be concerned that managing the system and training staff will not take excessive amounts of effort on their part (this is a major issue for the real ABC system). They will also want tools which will allow them to perform all the administrative tasks they need to do without being overly complicated.

Exam administrators (e.g. the exams office) will need the software to fit within their existing systems as much as possible, so the whole process is manageable. They will also be concerned about how exams will be invigilated, since there are technical issues on top of the traditional invigilation process.

Some people stated obvious functional requirements - “exam markers need to be able to mark exams” etc. rather than concerns. Others answered different questions entirely, such as what tasks these people need to perform.

d). State the four key principles of the Agile Manifesto (exact wording is not required, a paraphrase will do and they can be in any order), and for each one suggest a way it could be applied a project using the UP. (2 marks for each). (8 marks)

Value individuals and interactions over processes and tools. We could introduce some agile practices into the project in accordance with this principle, for example retrospectives at the end of each iteration to identify impediments and improve the way the team works for the next iteration.

Value working software over comprehensive documentation. We must make sure that we produce working software at the end of each iteration, and only produce other UP artefacts which are necessary, e.g. we will only produce fully dressed use cases unless there is a really good reason to.

Value customer collaboration over contract negotiation. It will be very important to talk to the stakeholders mentioned in question 2.3, to make sure that the software is fit for purposes, e.g. online exams can be fitted into existing exams office practices.

Value responding to change over following a plan. If, as a result of consulting stakeholders, it turns out that the software does not meet their needs, we need to find a way of fixing this.

Lots of easy marks here. The main reason people lost marks was by doing little more than restating the principle in question rather than citing specific agile practices.

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.

1. 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)

   a. Personally 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 is of course possible to generate questions by reverse-engineering the information in part b, but the problem with this is that you end up with questions which are too specific and 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

   Many people missed several of these domain concepts, in particular very few people had the notion of a device group, and several even missed the notion of a device. Adding human actors was ok, provided these key concepts were also there.

   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.

   Again, including human actors was fine, but omissions in the previous part tended to result in very incomplete diagrams.

   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 at 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 damned tricky.

   Most people got the first part right. However, many prioritised the management summary functions over the UI for the monitoring stations. The UI for people doing real-time monitoring is crucial (remember the Killer Robot!) and we’ll need feedback on that UI.

   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? (3 marks)

   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.
Some people came up with a Controller instead, which is definitely not the right thing 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.

A lot of answers didn’t even mention refactoring. If you didn’t remember what refactoring is (e.g. from week 4 of second year SE) is wasn’t very wise to do this question! Such answers got 1/4 irrespective of how good the descriptions of cohesion and coupling were.

b). Briefly explain what the two types of coupling are, and for each give a specific example of how it can be kept low in the exam software. Use a different example for each. (4 marks)

Internal coupling is between the classes which make up a subsystem. External coupling is between subsystems. [2] Examples: internal coupling is kept low in the classes representing questions by having an abstract Question class and subclasses for each kind of question; the subclasses are independent and rely only on the superclass. This could also be used as an example of external coupling, but to answer the question correctly we need a different example, e.g. model-view separation in the monitoring tool.

Although the internal/external distinction isn’t in Larman, I stressed it throughout the course and it was also in the second year, so there was no excuse for not knowing it and having to invent something else as several people did.

c). Suppose you have exam software which was originally designed for English language testing and was adapted to become general 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)

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).

A lot of people had a class hierarchy of results, rather than result generators, which doesn’t resolve the question of where the generation is actually done. Some used the adaptor pattern, which isn’t really appropriate here as there’s nothing to adapt to/from, we’re just generating spreadsheets in different formats.

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]

A lot of people failed to notice the second point - the original class would have needed to have a larger and less stable interface as a result of having to specify what kind of results are needed.

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. Most people got this right.

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 ResultGenerator subclasses. [1] The appropriate subclass instance to create would be determined by e.g. a parameter being provided or via a config file. [1]

Something like the third point above was necessary for full marks, to satisfy the “how” in the question.

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 okay.

Here, using the Adaptor pattern was more reasonable, and this and a number of other configurations were plausible enough to get the marks.

**Question 5**

a). Explain why design patterns are a tool for communication. Your answer should state who is doing the communicating and should use the Observer pattern as an example. (4 marks)

Design patterns give developers a common language for communicating design ideas. [1] Once two developers know a common set of patterns they can communicate very efficiently by using the pattern names such as Observer. [1] In the case of Observer the designers know they are discussing a flexible way of associating events (state changes) with code which needs to be informed of these events. [1] +1 for any other relevant point.

Many answers failed to make four clear points. On the other hand, there were many good answers which featured hypothetical conversations between developers which was a very nice way of answering the question A few people suggest that design patterns are used for communication with stakeholders, which is, to put it mildly, something that’s not going to happen very often.
b) The Strategy, Visitor and Template Method patterns address different aspects of the same general problem. Briefly explain what that problem is. (2 marks)

The problem of implementing complex algorithms in OOD [1]. By default an algorithm will be distributed over a number of classes defining an object structure, which can make it hard to understand. [1]

A lot of people just got this wrong, which I didn’t understand as it was stressed strongly both in the mini-lectures and in the notes.

c) Explain how the Visitor pattern could be used to help solve the graph matching problem discussed in question 1. Your answer should state whether the situation described is one where use of Visitor could be considered appropriate. (4 marks)

Note: since Q1 is compulsory, they have to read that description anyway.

First, yes it is appropriate, because the object structure (boxes, connectors etc.) is simple and stable, whereas the algorithms are complex and variable. [1]. We would define an abstract class, e.g. GraphItemVisitor with subclasses for each algorithm. [1] Within each of these classes there would be a visit method for each kind of graph items, hence collecting all the code for each algorithm in one class. [1]. Each graph item class, e.g. Box, need only implement a single method accept(GraphItemVisitor v) which calls back to the corresponding method in the visitor class. [1]

A number of people said no it isn’t appropriate because using Strategy as in Q1 is better. That may or may not be the case, but the scenario fits the criteria in which Visitor “could be considered appropriate” even if we ultimately don’t use it.

d). Explain how the Template Method class could also be used to help structure the graph matching code. (4 marks)

Template Method is a way of structuring code in any class hierarchy, so could be used along with either Strategy or Visitor. [1] The idea is that the methods which make up the public interface of the superclass are templates. [1] They provide common code (e.g., to match a box we might always want to get the text inside it) and then make use of abstract, protected methods to do the subclass-specific work (in this case the actual matching. [1]. The result is that programmers of the subclasses know exactly what they need to do - i.e. fill in the “slots” in the template and the compiler will even tell them this. [1].

Note: answers are likely to be more generic than this, which is acceptable so long as they are in some way related to the particular scenario.

Answers to this were generally good.

E) Briefly explain the idea of a Proxy (2 marks)

A proxy is a class which provides indirect access to another class, performing some sort of management of that “subject” class e.g. remote access, authorisation. [1] The proxy has the same interface as the subject, so the use of a proxy is transparent to the client code. [1]

A number of people missed the point about the proxy having the same interface as the real subject, which is important because it’s what distinguishes a Proxy from other kinds of indirection.

F) Suppose we want to construct a large number of variants of a multiple choice test, so that each student sees a slightly different version. A test is constructed from a large pool of MCQs held on a remote server, a small subset of which (i.e. those relevant to the topic of the test) will be used in each test. Of that subset, each MCQ will be used in multiple variants of the test. So for example there might be 100,000 questions in the pool, of which we use 50 altogether, spread among 1,000 variants which each have 20 questions. Explain how the Proxy design pattern can be used in an efficient design to meet these requirements. Hint: there are two proxy tasks here - briefly suggest whether there should be one proxy or two. (4 marks)

Firstly, we don't want to get the entire pool of questions from the remote server, so we need to get them on demand. Secondly, we don't want to fetch the same question from the remote server repeatedly. Therefore we need a remote proxy which will transparently fetch the questions, but we also want it to cache questions locally so once we've got a question once we can just use the local copy (a performance proxy). We could either have one proxy which does both, or we could have a chain of two proxies; the first looks for a local cached copy, and if found, returns it; otherwise it goes to the remote proxy to get it. An advantage of the latter design is that the two proxies are more likely to be reusable elsewhere.

A lot of people got hung up on access control, and hence access proxies. Although some access control would be required for this application, there’s no information in the question to say how/what is required in this area, whereas the numbers clearly imply a performance issue.

Some answers gave functionality to the proxy, such as having it select which questions to pick, which is not the job of a proxy. Some combined this with access control.