COMP33411 exam performance feedback 2014

Original marking scheme in bold, additional comments in bold italic.

On the scripts: / = 1 mark /2 = 2 marks X = wrong ? = dubious or unclear - judgement call whether to give a mark (usually not).

In general, there were a lot of very good answers, but also some very poor ones, where people seemed not to have engaged with the course at all.

Answer question 1 and two other questions. For full marks your answers should be concise as well as accurate. Calculators not allowed.

Question 1

This question is COMPULSORY

a). “Design patterns are a tool for communication”. Explain this statement, using an example. Your explanation should state who is doing the communication and what is being communicated. (4 marks)

Who = developers. What = best practices in software design. Experienced developers who know the patterns can just use the names. Experienced developers can communicate with less experienced ones by explaining the patterns, e.g. with UML diagrams. Full marks for any reasonable example e.g. “do you think these states are interesting enough to represent them as classes using the State pattern?”

Note: a common error is to answer in terms of communicating objects rather than communicating people!

Not a common error this year, but happened an number of times (also on 3a). On the other hand, there were a lot of very good answers, with good examples like that above.

b). In all organisations where the exam software is used, there is a requirement to store marked exam scripts for future reference. Suppose that this facility already exists, in that scripts can be stored in the format required by the University of Mancunia (UoM). However, the Royal Society of Double Accountants (RSDA) use a different format, and the University of Lochaber (UoL) use yet another, for example. Code exists to store the data in each of these formats, but the interface of that code is different in each case, and incompatible with the project’s existing code base.

Draw a UML diagram to show how the Adapter pattern can be used to solve this problem.. You should assume that the UoM storage code has a single public operation storeScripts() and that the storage code for each organisation can be represented as a single class which you should show on the diagram.. (5 marks)
The ExamSystem pseudo-class is optional, otherwise a correct solution should look almost exactly as above, apart from choice of names. An interface rather than as abstract class is also ok. The last part of the question is there because the version in Larman doesn’t show the adaptees.

Note that the UoM version doesn’t need an adapter, we can use the existing code directly.

For solutions which are roughly the right shape, marks deducted for:

- Not showing clearly the relationships between the classes making up the pattern.
- Adding clutter/making assumptions about storage methods which aren’t in the description.
- Abuse of UML notation.

A number of people interpreted the phrase “You should assume that the UoM storage code has a single public operation storeScripts()” as meaning that the operation belonged in the client class (ExamSoftware in the diagram above). I accepted solutions which made that assumption, provided the diagrams were otherwise consistent. Of course putting that operation in the adaptees doesn’t make sense - they will have different operations. Too many people lost marks for incorrect use of UML notation.
c). The **Facade** pattern has some similarities to Adapter. Explain what the similarities and differences are between the two patterns, and briefly discuss whether Façade would be a sensible alternative to Adapter for the problem of part b). (5 marks)

They are similar in that the both deal with the problem of incompatible interfaces [1] and they do this by placing a Pure Fabrication in the middle rather than changing any of the existing interfaces.[1] The main differences are that there is normally only one Façade, while there many be many adapters. [1] and that a Façade hides an entire system or subsystem, and can therefore be considered a last resort, while adapters are normally adapting a small number of operations. [1]. In the example, since we have a number of different interfaces to adapt to, a façade wouldn’t make much sense. [1]

*Most people answered this part of the question well, although there were a number of confusions, such as talking about “Façade controllers” or even use cases.*

d). Somewhere, there needs to be code that decides which adapter to create, depending on the organisation we’re storing scripts for. Where would it be sensible to put this code, and how might it make that decision? (2 marks)

In a Factory. [1] It could take a string representing the organisation, or read from a config file (mark for anything reasonable) [1]

*A wide variety of answers to this. A lot of people said in the abstract adapter. While this is possible, it would go against GRASP principles as it would reduce the cohesiveness of the class and also couple it to its subclasses, which is to be avoided whenever possible. I gave 1 mark for saying in the client class (ExamSoftware in the diagram above) as by Creator that’s the place you’d put it if you don’t want a factory.*

e). What are the consequences of using the Adapter pattern for this task in terms of cohesion and coupling. (4 marks)

Cohesion is good, each class represents a single well-defined entity with a well-defined role, e.g. the UoLAdtapter class contains code only to adapt from the UoM storage operations to the UoL ones. [1] Also, we’ve avoided cluttering up the UoM storage class with operations to store in other formats. [1] Internal coupling is low - the most coupled classes are the adapters which depend on the superclass interface (unlikely to change) and to the external classes (which might, but then the change is localised). [1] External coupling is very low - external code is coupled only to the superclass interface.

*A number of people made the classic mistake of not considering coupling carefully enough and saying it is increased.*
Question 2

This question was popular, and a “get out of jail card” for a substantial minority of students who seemed not to have engaged with the more technical parts of the course at all.

a). One key idea in the Agile UP is that all the UP artefacts that are not part of the final product are optional. In what other ways is the Agile UP agile? (4 marks)

It uses timeboxed iterations. It features regular contact with stakeholders. Requirements is a discipline which is expected to continues throughout the project, rather than a phase. We can make use of agile practices such as programming in pairs. Etc.

Many answers contained rephrasings of artefacts that are not part of the final product are optional. Others were just a list of agile practices (which may be used in the UP but aren’t mandated by, hence “can be used” in the standard answer above). User stories (UP uses use cases) cropped up a lot.

b). A slogan for agile software development is “embrace change”. One kind of change is requirements change caused by stakeholder feedback. What other kinds of change can happen (to requirements or otherwise) during a substantial software project? (4 marks)

e.g. A change of direction of the project due to institutional politics or funding. Changes of personnel in the development team. Need to conform to new legislation. (e.g. accessibility). Technical difficulties causing a scaling down of the project/opportunities causing a scaling up. Marks for any 4 sensible, distinct points.

Many answers contained changes due to stakeholder feedback - due to not considering a wide enough range of stakeholders - e.g. Beta testers are stakeholders.

c). 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?:
Many answers were in terms of requirements or concerns rather than risks. I ended up giving half marks where there was an implied risk, and marking part c as a whole /6, rounding up. However, I could instead have given zero to many answers.

i). Students (1 mark)

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

ii). IT support staff (1 mark)

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). They might not get VFM.

iv). Admin staff responsible for timetabling (hint: consider both short-term and long-term risks from their perspective). (3 marks)

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 considered “long term” to be next academic year.

d). 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).

The keyword here is “frequently” which in an agile method means ideally daily.

Students - no; the best way to address their concern is to ensure the admin staff can use the system effectively.

Many people said yes, ignoring the fact that creating a global timetable is orders of magnitude more complex than viewing an individual one. Of course it’s good if they are consulted, but that’s not the same as frequent interaction.
Uni management - ditto

Again, many people said yes, ignoring the fact that they will not be users. They will need to be kept up to date with progress, of course, but that’s not a frequent interaction.

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 accepted a range of answers to the IT support one, depending on what assumptions were made about the role of IT support.

e). One of the principles of agile software development is “Customer collaboration over contract negotiation”, leading to the agile practice of a customer representative working closely with the development team.

Why is the word “customer” not strictly accurate here? Hint: the timetabling scenario above is an example of the problem.

The customer is the person paying for the software, e.g. in the scenario above, university management. The customer is often not the person(s) who will actually be using the software - in this case timetable admin staff. Hence a better description might be “end user” or simply “stakeholder”.

Many people got this right. I gave one mark for saying there are a range of “customers, not just wrong. A number of people pointed out that students are the real customers via fees!

Question 3

This was the most popular question and often done very badly, particularly considering that the second half is another go at test 2.

a). It’s often said that design patterns are a tool for communication. Explain with examples why the same is true for GRASP principles. (4 marks)

Like design patterns they have names (High cohesion etc.) that can be used as shorthand by experienced designers. They can be used to discuss design tradeoffs (e.g. “by Expert, maybe we should move this operation to that class”). Since they represent best practice in OOSD, they are also good for teaching inexperienced developers, e.g. “that class has poor cohesion, it’s representing two different things, you need to separate them out.”

b). “Being agile, we Embrace Change. Therefore we should always design to maximise protection against variation”. How accurate is this statement? (2 marks)
We should design with PV in mind (many uses of inheritance do this for example) but there is a tradeoff - we can’s predict what will vary, so we need to be wary of spending a lot of effort on something which won’t happen, and of creating over-general designs that are hard to use in the simple case (like a lot of the Java libraries);

*I gave one mark for understanding PV and why it’s a good thing.*

c). 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.

*Almost everybody get this wrong. I made a point of it in a lecture and Larman says it too.*

d). 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 surprising number of people failed to come up with two correct answers. An abstract class to group similar parts (e.g. EnginePart) is not a PF because it is a domain concepts. A database is not a PF because it is not a class.*

e). Your junior developer has been reading about GRASP, so you are ready to set him another problem, about representing members of staff in the system. Different people use the system for different purposes, for example setters, markers, invigilators and admin staff. He comes back to you with:

"This looks like an obvious use of Polymorphism - we have an abstract class StaffMember and subclasses Setter, Marker etc."

Why is this "obvious" use of Polymorphism actually a mistake? (2 marks)

Because these are roles played by staff members, and one staff member can (and frequently will) be authorised to perform more than one, which would result in redundancy- the same real-world person represented by more than one software object."
f). Draw a UML class diagram which does use Polymorphism to represent this information given in part e), but in a more appropriate way. You should show the classes and the relationships between them, but no details of the internals of the classes (4 marks)

![UML Diagram]

For any marks a solution must be very similar to the above. For ones with the right shape, deductions for the same sins as in 1b.

A shocking number of people still didn’t get this even remotely right.

g). State two ways in which your design in part f) protects against variation. (2 marks)

If roles or added the rest of the design is not affected. We can change the way roles are represented without affecting anything else.

h). In practice, the problem is complicated by the fact that we not only want to restrict what staff can do, but also what they can do it to. For example a setter or marker only needs to set/mark on certain course units, while an invigilator may be able to invigilate any exam in the University. To solve this problem, we can make use of the fact that organisations like universities have a hierarchical structure. A University has faculties, which have schools which (for the purposes of assessment) have course units.

What data structure, making use of Polymorphism, would you add to your design to deal with this extra complexity? (2 marks)
We need to represent an organisation as a tree structure, where we can deal uniformly with arbitrary sub-trees. Hence we need an abstract class e.g. OrganisationTree, and either use the Composite pattern or have a fixed set of level. Either way, we would need to attach to each concrete role object a collection (e.g. a set or list) of OrganisationTree.

As per the feedback everybody got to test 2.

Question 4

Note: a significant part of this question requires answers in terms of actual Java code.

The 26 people who had the courage to answer this question were, with very few exceptions, richly rewarded.

a). The Strategy, Visitor, and Template Method patterns address different aspects of the same general problem. Briefly state what that problem is, and the role that each pattern plays. (5 marks)

The problem of representing complex algorithms in OO. [1] This arises because whereas in a procedural program an algorithms is normally localised in one place (e.g. a set of procedures within in module) in OO it is often distributed among several different classes, making it hard to understand, debug etc. [1] Strategy, by representing algorithms as objects, not only localises an algorithm in one class, but also allows us to switch between them at runtime. Template Method gives us a template for an algorithm which makes it easier to break it down into chunks. [1] Visitor gathers a distributed algorithm in one place, by inverting the normal OO structure.[1]

b). Show, in terms of actual Java code, how you would use Template Method and Strategy together to deal with the following hypothetical problem.

We are doing research into automatic marking of text answers. We have a number of different algorithms for this, but they all have the same structure. They first have a setup() part, which is always the same. Then they have analyse() and decideMark() parts which are different for each algorithm.

You should show the template method class, called MarkingStrategy (7 marks) and an example concrete strategy class called MarkingStrategy1 (3 marks)
You should assume that all methods take an Answer object as a parameter. Setup() and analyse return a modified Answer object while decideMark() returns an int, as does the main markAnswer() method. You should show the complete code for the template method Otherwise, indicate a concrete implementation of a method by a comment
Minor errors in Java syntax will not be penalised, but for full marks you should show correct visibilities as well as what is abstract and what is concrete. (10 marks)

```java
public abstract class MarkingStrategy {

    public int markAnswer(Answer inputAnswer) { // main method is concrete and public. [1]

        Answer answer1 = setup(inputAnswer); [2 for something approximating to this]
        Answer answer2 = analyse(answer1)
        return decideMark(answer2);
    }

    private [1 - half for protected] Answer setup(Answer ans) { // Setup method has a concrete implementation
        // which the subclasses don’t need to know about.
        // Concrete implementation here. [1]
    }

    protected abstract Answer analyse (answer ans); // The abstract methods are protected,
    protected abstract int(decideMark) (answer ans); // The public interface is markAnswer; [2]

} MarkingStrategy

public class MarkingStrategy1 extends MarkingStrategy [1] {

    protected Answer analyse (answer ans ) {
        // Concrete implementation here. [1]
    }

    protected int(decideMark) (answer ans ) {
        // Concrete implementation here. [1]
    }
} MarkingStrategy1

Most answers were along the right lines. Main errors were in visibilities - the abstract methods can’t be private! -and having Setup implemented in the subclasses.

c). A function of the exam software is to produce reports. However, these reports may be in many different formats, which we can’t anticipate in advance. Outline how the Visitor pattern could be used to scan a data structure containing answers to generate a report. (3 marks)
We would have an abstract Visitor class, e.g. ReportVisitor with subclasses for each kind of report. Within each visor class we would have a visitor method for each kind of answer, e.g. visitTextAnswer(TextAnswer ans) Then each class to be visited needs only to have a single extra method, accept() which is a callback to the corresponding visitor method.

Most answers were at least along the right lines.

d). Is this likely to be an appropriate use of Visitor? Hint: we don’t add new answer types very often. (2 marks)

Well it’s what I actually did. Yes it’s appropriate because the object structure being visited is stable (we don’t add new answer types very often) but the algorithms (report generation) is unpredictable.

There were a number of confusions here, but most people got it at least partially right.

Question 5

Only 16 people did this question. The material is new this year so there are no past papers, and I probably didn’t teach it very well. I therefore marked it generously.

a). Explain, using an example from the exam software, the problem which Aspect-Oriented Programming is designed to solve. (4 marks)

The problem of cross-cutting concerns - ones which cut across the natural class structure and are hard to localise within one class. For example logging is a requirement for many operations in ABC. Writing code of the form doIn(); logIt() is uncohesive and couples many classes to the logging API.

Most people got this more or less right.

b). Briefly explain what the two main constructs within an aspect in AspectJ are, giving an example of how they could be used to deal with the example problem of part a). Accurate AspectJ syntax is not required provided the meaning is clear. (6 marks)

A pointcut marks a well-defined point in the execution of a program. To solve the logging problem we could have a marker interface Loggable inherited by all classes whose method calls need to be logged, and have a pointcut something like:

pointcut loggedOperation execution(*,*) & & this (Loggable) (i.e. any operation on a Loggable). [2]

Advice is code to be executed when pointcut is reached. [1] In this case we need this to happen after the call:
after() loggedOperation(): logger.log(...)

Likewise most people had something which looked close enough.

c). State the two main reasons why attempts have been made to improve the implementation of design patterns using aspects, and what AspectJ constructs are required in each case. (4 marks)

The first is that pattern code (e.g. the list handling in Observer) can get tangled with business logic code. Pointcuts and advice (as above) can deal with this. The second is that many patterns require inheritance, and in a single-inheritance language such as Java, there is the possibility that this clashes with a nature inheritance hierarchy in the domain. This can be worked around with the declare parents construct.

There was a lot of confusion about this. I don’t think anyone mentioned declare parents.

d). Is the use of AOP likely to improve an implementation of the Adapter pattern? Briefly explain why or why not. (3 marks)

No, because the Adapter is a Pure Fabrication which does not have any functionality or inheritance relationship in the domain. Therefore we would just be moving code from the adapters into aspects, leading to a less clear design.

Again, few answers gave a clear explanation here.

e). State three disadvantages of AOP which should be taken into account when deciding whether or not to use it on a project. (3 marks)

3 of: All developers, present and future, need to be familiar with it; it constrains what tools we can use; overuse can lead to poor design; to fully understand any code which uses aspects, you need to look in two different places.

Most people made three relevant points.