Module Title: Software Architecture, Process and Management
Exam Diet (Dec/April/Aug): April 2014
Brief notes on answers:

1. (a) You could choose GPES here and the engine management system architecture we saw in the section on product lines. Provide outlines of the two architectures focussing on the QAs you are concerned about. Here I think they have contrasting performance and availability requirements but similar testability or modifiability requirements. You could choose several other possibilities.

   (b) Choose modifiability and set up a scenario that requires a relatively slow turnaround on modification because of security requirements for GPES and safety requirements in the case of the controller system.

   (c) Choose performance and set up a scenario that requires real-time performance. The controller architecture is set up to achieve this while nothing about GPES is concerned with real-time.

   (d) You should observe that there is no feature in GPES that is intended to manage resources to achieve real time whereas that is an important feature of the controller architecture.

2. (a) Probably MVC is the best match - the model captures offers, needs and matches, it has different interfaces and controllers for each user class, etc. Two marks for the choice and two marks each for justification of how it matches the structure of the problem.

   (b) Blackboard is poorly matched because the problem structure does not fit well with the "problem solving" decomposition approach of blackboard and there is no nice way to represent the state of the world in a blackboard approach. Two marks for the choice and two marks each for justification of how it doesn't match the structure of the problem.

   (c) This should be an MVC diagram where the model is annotated to carry the needs, offers and deals and there are three view/controller pairs for the different users. Allocate two marks for the diagram and up to three for the annotation.

   (d) There are several possible tactics. For example choose *authenticate users* and *maintain data confidentiality* the first would require some additional authentication features in the architecture that require all access is via authenticated users. Data confidentiality would require some components that managed encryption and the addition of some means of allowing control over encryption keys etc.

3. (a) Here you could choose some suitable fixed architecture that allows easy reconfiguration e.g. some sort of layered architecture and a tiered model of runtime/deployment and consider this against an product line approach that looks at trying to rationalise the product offering by looking at taking a more thorough approach to componentizing the system.

   (b) One approach is incremental the other is more radical so the balance is risk against benefit. The potential benefits of PLA are quite high but it means a radical shake up of the approach to product. Some more modest re-architecting is easier to risk assess but will probably have less benefit..

(c) Here you might consider some modification to the development process e.g. spiral would be more risk-based and so you might expect to see products abandoned earlier if they are not going to work or be too risky; I'm also looking for some mention of devops and the potential there for much faster introduction of changes to systems.

(d) Something like a spiral model would require some rebalancing of the relationship between customers and developers because of the focus on risk (similarly if you chose something agile here) customers would need to be involved in assessing the risk of progressing etc etc. In the case of devops you would see a radical change in relations between development and operations stakeholders - working more cooperatively on resolving issues of early release versions. Taking more of a share of risk across development with operations helping to identify and resolve issues.