UG Exam Performance Feedback
Third Year - Semester 1

COMP 31111 Verified Development

Verified Development

Richard Banach

Comments:
Q1: Very well answered on the whole.
Q2: Very well answered on the whole.
Q3: One answer; somewhat confused.

COMP 32111 System-on-Chip modelling with System C

System-on-Chip modelling with System C

Steve Furber
Alasdair Rawsthorne

Comments:
From: SBF

Q1: (Compulsory)
   a) generally good answers, but several candidates missed "every 2 years"
   b) some confusion here - check notes more carefully!
   c) few candidates observed the key issue of circuit-switched vs packet-switched
   d) switching thread to hid latency generally understood, but many missed the
      need to note importance of absence of dependencies between threads.
   e) generally OK, but not all noted that "dynamic" means "on the fly" (or equivalent)
   f) generally good, but some gave general, not Transactional Modelling-related answers
   g) distressing numbers of students seem to be unable to work out how to solve this
   h) mostly good, some didn't appear to realize that header files and libraries were
      involved
   i) generally good
   j) qualitatively answered well. Estimates (based on students' memories and impressions
      of lab work) were all over the map.

Q2: The question was attempted by half of the candidates. Most made a decent job of it, but
a few failed to address the key issues, and a common error was to consider reconfigurable
logic more flexible than a general-purpose CPU. While it is true
that an FPGA can be configured to do almost anything, flexibility should take into account
the overhead of configuration, which is significant. GPCPs are "universal" with no
overheads to switch from one function (program) to another.

From: AR

Q3: The other half of the student group answered this. As a result of their extensive
experience of Algorithmic, untimed and timed-TLM modelling, and our emphasis on
correctness in the lab, most gave very credit-worthy answers. No systematic holes in
understanding appeared.

Q4: not a single student answered this.

Q5:

Most students tackled this, with most of these giving good answers. Common omissions
were to fail to relate arbitration and routing to TLM; implying (rather than stating) behaviour
of the code provided, and (particularly rampant) logic and timing errors in modifying the code
to give somewhat unconventional priority behaviour.
As a general comment and similarly to previous years, simple book-work questions proved to be surprisingly challenging, as opposed to questions asking for ‘practical’ examples.

Question 1. This question was taken by almost all students. In part (a), a number of students failed to describe what refreshing a data-warehouse means in terms of approaches (incremental, full) and refresh-rate. The first bit of part (b) was generally OK, although a number of answers were defining support and confidence rather than saying how these are used for finding interesting rules. The second part was relatively easy (although some answers did not specify the support for the rule). In the third part, surprisingly many errors were in using support as an absolute number (and not as percentage) whereas confidence was used in percentage, leading to wrong results. Part (c) did not ask for description of decision trees, but rather for comparison. Average mark for this question was 10 (ranging from 2.5 to 19.5 out of 20).

Question 2 was taken by 75% of students. Part (a) required understanding as what “unsupervised” meant in terms of training data, exploration etc, and some students struggled with that (including answers that data analysts need to be supervised). Part (B) included a typical bookwork question (i) and a relatively simple application of that to the example given. Part (ii) required explanation: acceptable answers included descriptions of entropy etc and/or showing what would happen if you split the tree on a particular attribute. Part (c) was discussed in the class and required explanations of variability (no extra marks for discussing ambiguity). The average mark for this question was 12 (ranging from 7 to 20).

Question 3 was also very popular (taken by most students), with an average of almost 12 marks (ranging from 0 to 20). Part (a) was mostly bookwork, but with some students completely missing the point (sampling is not aimed at producing false positives(!) and frequent itemsets are not sampled just randomly). There were no wide problems with part (b) for many students, although occasionally there were new ‘creations’ of k-means that were not correct or the distance measure was not applied correctly (e.g. using Euclidean instead of max). The task included calculating only one (next) iteration, so no marks are given for providing 2 or more iterations. Part (c) was generally ok, although a number of students failed to define proper dimension tables.

Question 4 was not a popular one (one quarter of students), with an average of 11 (range 6-20). Part (a) was bookwork and not many problems were spotted. Part (b) was also mostly bookwork, but some students managed to confuse inter-cluster distances with distances between data points (e.g. Euclidean, cosine, etc). Part (c) required some discussions about the different approaches that have been mentioned in the question. Some of those were relatively vague (“it is a difficult task”), without giving any specific reason for that.

Question 5 was also not a popular choice, with a low average mark of 98 (range 1.5 ot 17.5). Part (a) was reasonable, although many answers failed to explain what ontologies are and – most importantly – how can they be used as meta-data. In part (b.i), many answers struggled to explain the CUBE/ROLLUP options. Part (b.ii) was a disappointment, as many students struggled to write a reasonable SQL statement (even without CUBE/ROLLUP extensions). Many answers included CUBE instead of (correct) ROLLUP, and some marks were lost there. In part (c), there were answers that did not talk about data mining; still, a number of students proposed reasonable features to be used for the task.
COMP33411 January 2012 Exam performance feedback

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

Generally the answers were very good and showed a significantly better understanding than at the corresponding tests. Well done!

Question 1

1.1: Explain the notion of a design pattern, and why design patterns are important in object-oriented software development. 4 marks

Bookwork

This was answered well by almost everybody. A few people confused design patterns with other things, including, bizarrely, domain models. Any answer which included stakeholders was wrong! The most common omission was probably pattern names, without which it’s hard to explain why design patterns aid communication.

1.2 Suggest ways in which the Observer design pattern could be used in the software, particularly in the implementation of the Take Exam use case. 4 marks

In the GUIs, in the form of the Java delegation event model. More fundamentally to organise the communication between the clients and the exam server. Either way round, or both, is plausible: the clients can be observers, registering themselves with the server and being notified of events (in fact we do this, but only for the special case of sending the start signal) or the server can be an observer, looking for changes of state in the clients (in fact we do most things this way round, but strictly it’s not the Observer pattern as the server is fixed at client startup time). Marks for any reasonable suggestion. Generally this too was well answered, with many suggestions along the lines above. A number of people suggested using it for loss of focus monitoring, which is actually the one places we really do use it. Marks mostly were lost due to not knowing the Observer pattern and/or not explaining it coherently (it was hard to tell which in some cases).

1.3 One of the very earliest design decisions in the real project was to use the Composite design pattern in the core data structures of the application such as Question and Answer. Explain why. 4 marks.

Questions may have sub-questions and sub-sub questions etc., so we need a notion of composite questions which contain other (possibly composite) questions. Atomic questions would include text, multiple choice etc. A method such as getMarksAllocated() would be declared abstract in the abstract Question class, and would be implemented in CompositeQuestion by recursively getting the marks from the subquestions, and in AtomicQuestions by just returning a stored value. The same structure also applies to Answers.

Marks also for relating this to high cohesion/low coupling, PV etc correctly as many people did.

This was a very easy question and so was marked quite strictly. Nevertheless a lot of people got full marks. Marks of 3/4 were generally due to not saying four distinct relevant things.

Lower marks were generally due to not understanding the Composite pattern, sometimes confusing it with other patterns. A number of people said that it allowed Questions and Answers to have common operations, which is wrong - they form separate class hierarchies and will have very different sets of operations.

1.4 Draw a UML diagram to illustrate the Composite pattern as applied to Question and related classes, showing one important operation.6 marks

The following is all that was required for full marks:

Acceptable variations included:
• Showing the operation in all classes (I prefer not to as if an operation is declared abstract in the superclass, by definition it must occur in the subclasses)
• Showing an operation specific to one subclass (e.g. addQuestion() in CompositeQuestion)
• Adding additional relevant classes, e.g. ExamPaper.
• Including an AtomicQuestion class as a superclass of TextQ and MCQ
• Having a different set of atomic questions.
• Adding a small, relevant, set of key attributes.
• Including Answer and subclasses (not my intention but a plausible reading of the question).

However most solutions which did this were broken for other reasons,

I was shocked at the quality of answers to this. Many people simply could not reproduce the Composite design pattern, and many others failed to follow the instruction “showing one important operation.” The number of marks lost due to failing to use the notation correctly was less than in the tests, but still significant.

1.5 Suggest how two other GoF design patterns might be used in implementing the Take Exam use case. 2 marks

e.g. in a randomised multiple choice test the questions could be generated from a Factory which would be a Singleton. The exam client could have a Façade controller. Strategy could be used for different end-of-test behaviours for different types of assessments. Marks for anything reasonable.

Answers to this were of mixed quality to say the least. Several people answered the different question “… used anywhere in the exam software”. This was particularly prevalent among the longer answers - typing lots is pointless if you don’t read the question properly. Other common errors were saying that the product of a Factory is a Singleton rather than the Factory itself and answers which just didn’t make sense, particularly ones which mentioned Visitor.

A number of answers suggested Controllers (either UC or Façade). This is a tricky one because Controller is not actually a GoF pattern, it is “officially” a GRASP pattern so strictly speaking such answers are wrong. However, it is in my view misclassified, and it clearly could be used in the application (in fact we don’t because the client is too simple to need it). So this gave be a dilemma, which I resolved by allowing Controller iff it was clearly explained (clarity is more important than classification).

Question 2

a) State four key principles of Agile software development. (4 marks)

The easiest way to answer is to quote the agile manifesto. Value:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation (or “minimise ceremony”)
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Most people did exactly that. Those that didn’t tended to get unstuck by saying the same thing (typically number 4) multiple times, or quoting practices rather than principles.

b) Give one example of how each principle is reflected in agile practices (4 marks)

1. There are many agile practices which reflect this, for example stand-up meetings encourage positive interaction between team members.
2. Working software is produced during each iteration (sprint). Comprehensive documentation isn’t
3. A customer representative is (ideally) a full member of the project team.
4. e.g. in SCRUM the team measures its velocity during an iteration and adjusts the priorities for the sprint accordingly. Or responding to customer feedback at the end of each
iteration.
Most answers for this were reasonable, but tended to be more detailed restatement of the principles rather than specific practices. I allowed this to some extent but for full marks you needed four specific practices. Also, several answers were excessively long - the question says “Give one example…”

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

A good answer is:
The UP, is like Agile methods, iterative with working code produced at the end of each iteration. [1] It specifies many artefacts which can be produced, but all, except the code, are optional; hence the UP can be applied in a Agile way. [1]

I accepted a variety of answers to this, although a number of people failed to make two separate coherent points.

b) Do Agile UP and Scrum project teams differ significantly in the way they interact with stakeholders? (2 marks)

In general terms, no - both emphasise stakeholder involvement and feedback [1]. Scrum takes this to the limit, with stakeholder representative working full-time with the team - the UP does not mandate this. [1]

Surprisingly, many people got this wrong. Some people even thought that Scrum teams only interact with the stakeholders at the end of development!!!!

d). State two similarities, and two differences, between use cases and user stories. (2 marks)

They both capture functional requirements 1. They are both written in the users’ language. 1. Uses cases can be written at difference levels of detail, user stories have a fixed, brief, format. 1 User stores are written on physical cards, UCs are normally written electronically. 1

Marks for other correct points provided there are two of each.

Half the people answering this question knew what user stories are and half didn’t and this was reflected in the marks. More worryingly, a number of people confused use cases (which are textual descriptions) with use case diagrams (which are an overview of a set of use cases and not relevant to this question).

e). You are tasked with introduction of the agile UP into a large organisation that produces computer software for its internal needs. What arguments would you use to persuade non-technical senior and middle management that this is a good move for the organisation.? (4 marks)

The challenge here is to justify the agile UP in non-technical terms. Some good points to make are: Agile is now mainstream, having been around for more than a decade - so is the UP; an important aspect of Agile is embracing change, so the software development will be able to keep up with the changing requirements of the organisation; agile emphasises teamwork, and self-improving teams which not only makes effective use of staff time but is also good for morale (making it easier to hire high-quality staff. The Agile UP in particular encourages risk-driven development, where high-risk aspects of a project are talked early.

Most people did a good job of this, considering it was done under exam conditions (the wording of many answers would have needed tightening up if you were talking to real managers). People largely avoided technical terms or explained the consequences of them.

Question 3

a). Explain, using an example, how a domain class diagram can be used to gather useful information from stakeholders. Your answer should take into account the different kinds of
skills which different stakeholders have. (5 marks)

An initial domain class diagram can be drawn after an initial interview with a stakeholder, to identify important domain concepts and understand the stakeholder’s language. The diagram can then be inspected to identify areas of uncertainty and generate further questions for the stakeholder (multiplicities are a particularly good source of questions). An example from ABC is that that there is a requirement for two or more invigilators per room, not per test. For stakeholders with an appropriate technical background, it may be most efficient to show them the diagram with areas of interest marked. Otherwise the diagram should not be show, but is list of questions arising from it should be prepared in advance.

This was a tricky one. As the above marking scheme suggests, I was thinking of technical vs. non-technical users when I referred to “skills” (something I spent some time talking about in the mini-lecture but isn’t on the slides), but may people interpreted “skills” as the different roles people play in using the exam software - students, markers etc. I accepted answers based on this. However for full marks it was necessary to answer specifically about “how a domain class diagram can be used to gather useful information from stakeholders” rather than just saying why domain modelling is a Good Thing. A few people failed to provide an example (even though in the online version there was a source of examples directly above on the screen!

b). In the UK many professional bodies (e.g. the National Institute of Advanced Plumbers, the Royal Society of Double Accountants) conduct nationwide paper-based examinations for thousands of students at a time, conducted at test centres across the country. Some important aspects of the process are as follows:

“Exam papers are prepared, and sent to the organisation’s central administration facility, where they are copied and sent out to all the test centres by courier. All the students sit the exam at the same time. Once the test has been taken the student’s exam papers are copied at the test centre, and copies are sent to two markers who mark independently. Both copies of each paper are then sent to a third, senior, maker who resolves discrepancies between the marks, and in doing so also checks the performance of the other markers. Finally all the papers, and the marks, are returned to central admin, who then inform the students of their results. At all stages of the process, rigorous procedures are in place to ensure that the right people get the right pieces of paper.”

Draw a class diagram to represent the significant domain classes suggested by this description and their important relationships and attributes. 7 marks

At this level of description it makes sense to include the actors, and also the notion of a test centre. Question and answer papers are essential (the ambiguity of “exam paper” in the question is deliberate). The fact that two markers and one arbiter are needed for each paper should be clear from the diagram. (Generalisation with an abstract Marker class would not be silly).

Answers to this kind of question commonly include cluttered domain models with too much detail, hence full marks only for a clear and truly informative diagram.

There were a lot of reasonable attempts at this, but none good enough for full marks. In particular, only one person resolved the ambiguity of “exam paper”. Some answers missed important classes such as “test centre” while others were cluttered with extraneous attributes. Many people represented the two initial markers as two separate classes rather than simply using a multiplicity of 2 (representing the senior marker as a separate class does make sense as their role is different). Nevertheless, overall the answers were reasonable considering the time pressure.

4.3 if we were to replace this paper system by an electronic one, to what extent would these domain classes correspond to design classes? 4 marks.

Some cases are clear – there would definitely be software classes representing question and answer papers for instance. The others would depend on the scope of the system and whether should be design classes such as TestCentre would require further investigation. There would be far more design classes than the small number of domain classes suggested by the high-level description above.
A couple of people spotted at this stage that separate QuestionPaper and AnswerPaper classes are required! A lot of people failed to make four separate points. A number of answers were too generic, talking more about the correspondence in general, rather than for the particular scenario. Some parts of some answers would have been more relevant to 4.4.

4.4. Suggest ways in which the process could be improved in the software implementation. (4 marks.)

Obviously we eliminate expensive and error-prone copying and moving of large amounts of paper, and can simplify the procedures associated with that. The clunky triple-marking process could be streamlined. Marking would be much quicker, and hence less markers are needed, reducing the (very real) danger of having to deal with dodgy marks. The software could also do sanity checking on the marking process to improve consistency, and even automatic marking in some cases. Collecting data about e.g. regional variations in performance, would be far easier. Etc. etc.

This was generally well answered, except that a number of people again failed to make four separate points.

Question 4

a) Briefly explain the role of GRASP patterns in object-oriented software development. (3 marks)

A good answer is

They provide a set of principles for assigning responsibilities to classes, the most difficult skill in OO software development. [1] Like all patterns they form a language which helps developers to communicate [1]. They form the “building blocks” for design patterns. [1]

Answers with examples are also good.

Over half the answers to this got full marks. Those that didn’t, in almost all cases, simply failed to make three separate, relevant, coherent points.

b) Explain the GRASP principles of Polymorphism and Protected Variations, and how they are related, using examples from ABC. (6 marks)

Polymorphism – being able to send the same message to different types of objects and have them respond appropriately (implemented via inheritance). [1] Protected variations – abstract the Thing Which Varies, so when the Thing Which Varies varies, it doesn’t trash the rest of the code. [1] Often achieved via polymorphism, as having an abstract class decouples the rest of the code from the concretet subclasses [1] e.g. in ABC it allows us to add new question types as subclasses of Question without changing existing code. [1] PV can be achieved without Polymorphism, e.g. choosing fonts which will work across platforms. [1] Polymorphism only achieves PV if it is used wisely however (e.g. used to model roles it has the opposite effect [1].

Other sensible points also get marks.

Since it’s easy to make three separate points on parts 1 and 3, this was designed to be a challenge to get full marks. Most answers got at least 4/6 though A number of answers were confused because they failed to recognise that Polymorphism (like e.g. a Controller) is something you use, while PV (like e.g. high cohesion) is something you try to achieve. In general almost everybody could describe Polymorphism coherently, but there as a lot of confusion about PV.

c) Briefly explain the GRASP principle of Indirection, and how it relates to Polymorphism and Protected Variations (3 marks)

We add indirections to decouple one subsystem from another. [1] For example by using an abstract superclass we decouple external code from details of the subclasses. [1] Indirection promotes PV because it allows parts of a system to vary independently. [1]
I expected a lot of people to struggle with this as Indirection did not appear on any of the tests and was not emphasised in the mini-lectures. This was indeed the case, but there were also quite a lot of good answers. In particular, several people mentioned Proxy or Adapter, which are good examples of Indirection.

d) We have a tool, implemented as a Java applet, which monitors exams in progress by accessing information from the server every few seconds. It contains a table with a row for each student, showing the status of the student, for instance how much time they have left, and when their work was last backed up. The invigilator has various display options, such as showing only student currently working, or ordering the table based on any of the columns.

Explain the notion of a Controller, and the different types of controller, using the monitoring tool as an example. Suggest what sort of controller, if any, would be most appropriate here. 5 marks

A controller is the first object beyond the GUI layer which controls system operations. A façade controller represents the overall “system” or “root object”, or a specialised physical device (not relevant here). This makes sense if we think of the tool as a message handling system, communicating with the server in an asynchronous way. A use-case or session controller represents the control required to manage a use case. This makes sense if we consider the primarily as stepping through the sequence of steps to monitor an exam. Such a controller could be based around the Invigilate Exam use case. It’s debatable whether a controller is really necessary here, as user interaction is (deliberately) simple.

This was generally well answered. Most people knew the answer to the first part and said sensible things about the second. Most people opted for a Façade controller, which fits with the description, which describes a set of features rather than a use case. A number of those who opted for UC controllers did so on the basis of UCs which were too small - e.g. reordering the table is not a good UC as it doesn’t pass the “Boss Test (as discussed in COMP234).

e) How might other GRASP patterns be applied in the design of the monitoring tool? 3 marks.

The Observer design pattern is strongly indicated, hence polymorphism.

Marks for anything sensible, e.g. Pure Fabrication could be applied to handling of messages between tool and server. Model-view separation -> low coupling. Etc.

Answers to this part were very disappointing. Lots of people just listed GRASP principles without relating them to the monitoring tool. A number of people listed GoF patterns rather than GRASP - which is only ok if they are related back to GRASP (as in the example above).

Question 5

a) Explain why design patterns are a tool for communication. Your answer should state who is doing the communicating and should include two specific examples of design patterns. (4 marks)

Design patterns enable quick communication between experienced developers They also allow experienced developers to communicate best practice to inexperienced ones. An example of the former is “do you think we should use a Factory to create the Adaptors”? An example of the latter is that you might draw a class diagram to show the Strategy pattern, and giving an example of its use.

Most people got full marks for this. Some people named GRASP principles rather than design patterns. I allowed this provided at least one actual design pattern was mentioned.

b). Explain the idea of a Proxy, and give two difference examples of proxies which could be used in the exam software. (3 marks)

A proxy is inserted between some using code and some resource (or “subject”) in order to manage that resource in some way. [1] A Remote Proxy manages remote communication and could be used to manage client-server interaction in the exam software. [1] An Access
Proxy controls access to a resource and could be used to authenticate staff users with different access rights. [1]

Most answers to this were very good (although some were unnecessarily long). A number of answers talked about performance proxies rather than access proxies, which was fine. A few people lost a mark by not actually saying what a proxy is (although giving valid examples)

c) Briefly Explain how the Visitor design pattern works (4 marks)

Visitor allows us to manipulate an object structure with code outside of that structure. A visitor class has a method visitC for every class to be visited. Each such class has a single extra method accept(visitor) which runs the relevant visitor method. An object structure may have multiple visitors, in which case there is an interface which all the Visitor classes implement and the classes in the object structure still need only a single accept method.

Answers to this varied widely, depending on whether people actually knew the pattern or not. A few answers were so badly written that it wasn’t possible to decipher them reliably.

d) Under what circumstances is it appropriate to use the Visitor pattern? (2 marks)

Visitor inverts the normal OO structure and collects the whole of an algorithm in one class. This is appropriate when the object structure is stable but the algorithms to be applied to it are complex and liable to change. Otherwise it’s best to make each class responsible for its part of the algorithm in the normal OO way.

About half of the answers were fully correct. Most of the others at least had the general idea that Visitor should only be used in particular circumstances.

e) Briefly explain the idea behind the Flyweight pattern (2 marks)

Flyweight is an optimisation which avoids the creation of large numbers of small objects. [1] It works by replacing intrinsic state within an object with extrinsic state stored in a separate data structure. [1]

Everybody had the right general idea. However, there was a lot of confusion about extrinsic vs. intrinsic state, and a number of people thought the “flyweight” held the extrinsic state, which of course is the opposite of what needs to happen.

f). How are the Proxy, Visitor, and Flyweight patterns related to GRASP principles? Name five different GRASP principles in your answer.

A Proxy is a Pure Fabrication and an Indirection which reduces Coupling and improves Cohesion (as the management function is separate from both the subject and the code using it) it makes use of Polymorphism as the Proxy and the subject present the same interface to the using code via an abstract superclass.

Visitor is much less obviously related to GRASP, hence the safety warning. In its most general form it does make use of Polymorphism, and arguably it improves cohesion if used wisely (it’s hard to argue that it improves coupling).

Flyweight is an optimisation and not directly related to GRASP (it actively increases coupling). However flyweights are created from a Factory which is a Pure Fabrication designed to improve Coupling and Cohesion.

(Full marks for any fully correct answer which mentions all three patterns and five GRASP principles)

Half the answers for this got full marks. Those which didn’t mostly named five GRASP principles but didn’t relate them to the patterns correctly (or in some cases, at all).
Question 1 (AFC)
Insufficient answers to give meaningful generic feedback

Question 2 (AFC)
Main problem with the answers to this question was for part c). The question is asking for the definition of a meta-model that can be used to define customised solutions by end-developers. This means that the model must define generic attributes and relationships that can be used by the end-developer to create specific ones. The answers made decisions about what attributes would be allowed and prevented end-developers from defining the ones that they needed.

Question 3 (SME)
This question aimed to test understanding of basic concepts about what makes for a good user story. There were many excellent answers, although only a few candidates managed to identify all the problems in all of the stories.

The most common misconception that resulted in lost marks was the idea that ambiguity in user stories is always a bad idea. This notion probably has its basis in material taught in other courses on more conventional approaches to requirements gathering. When writing use case scenarios, for example, we would definitely try to be as precise and clear as possible, and to avoid ambiguity.

When performing agile requirements gathering using user stories, however, we take a different line. We accept that we can never remove all ambiguity from a requirement written in natural language, and we do not try to. Remember that a user story is *not* a statement of a requirement, it is a reminder to have a conversation about some aspect of the system's functionality. Ambiguities don't matter so much in a user story, since we can clarify them during this conversation. In fact, the ambiguity that natural language allows is positively embraced by user stories as a major benefit. It is this ambiguity that allows us to specify requirements in a just-in-time, just-enough fashion. Stories that are a long way from implementation *should* be written in an ambiguous way, while stories that are close to implementation should be more precise (as far as is easily achievable in the time available).

Why, then, did I emphasise the importance of getting accurate user roles in stories? Isn't this just another example of ambiguity in a user story? It is, and we could legitimately imagine writing stories initially with vague user roles, and refining these into more precise ones later on in the development - this is precisely what split-by-subset-of-user does, for example. The point I was making, though, is that while some kinds of ambiguity are helpful in agile (or at least harmless), some kinds are actually harmful. This is the ambiguity or vagueness that arises through laziness. There is a more precise word that we can immediately see is appropriate, but we don't use it because we don't do the 2 minutes of thinking that would get us there. The role in the first of the stories in this question is of this type. Just saying "user" is just lazy. Even the bland "customer" would have been an improvement. But what is wrong with "concert goer"? "Music fan"? Even "potential ticket purchaser", if we want to emphasise the business role? These roles are quick to come up with, and help us more in generating further useful stories than the bland role given here.

Candidates lost marks, therefore, for pointing out examples of useful or harmless ambiguity and saying they were problems in the stories. For example, a number of candidates complained about the use of the word "quickly" in one of the stories, as it was "ambiguous". But it is not clear to me that there is a more precise alternative that would provide much more information, is there?

A number of candidates complained that certain words were not suitable for use in user stories. "say that" was one example, and "create" was another. I don't know what the perceived problem with "say that" was. Plain and simple language is generally to be preferred to more complicated language in most communication settings. The latter misconception, regarding "create", obviously came from a discussion during the revision session when we talked about not using the standard CRUD operation words when a more-domain specific term was available. That is, we shouldn't say "create an account" if the
customers all talk in terms of "registering an interest". But, there is nothing intrinsically wrong with the phrase "create an account", or the word "create" in itself. If that phrase/wording is one used by the customer, then it absolutely should be used in the stories you write for them.

A few answers claimed that story (vi) was a compound story and that compound stories are bad. In fact, the problem with this story is that its goal is poorly written, and takes the form of a function rather than a description of actual business value. A true compound story has multiple functions crammed into a single story function phrase. There is nothing wrong with compound stories - unless you are about to implement one, in which case you should refine it down to the individual stories pretty sharpish.

Finally, a small number of answers said that certain of the stories were epics and "therefore not user stories". This took me by surprise. An epic is just a kind of user story (a "big" one). All epics are user stories, although not all stories are epics.

B) Split-by-operation is the easiest of the refinement strategies in general, largely because it is so similar to the forms of refinement we are familiar with from conventional requirements gathering. This was a fairly straightforward refinement task, and many candidates gave excellent answers. One of the biggest causes of lost marks was a failure to refine the business value for the new refined stories (i.e., the refined stories just had the same business value as the original one).

Another issue that came up quite frequently was candidates working hard to avoid using the CRUD words, when no more precise domain alternative existing. Instead, bland synonyms for the CRUD terms were used, that added nothing to the precision of the story. E.g. a number of candidates used "modify" and "alter" for "update", when the real problem with writing a good update refinement for the story was in specifying precisely what was updated. I did not penalise the use of these synonyms, but I did award marks to candidates who pointed out that the update refinement was not really appropriate (since updates to course choices are more normally done by deleting one course choice and adding in another).

C) The number of good answers provided to this question was far fewer than for part b), reflecting the more challenging nature of the split-by-subset-of-user refinement technique. Most people managed to come up with sensible role subsets (team member, project marker, etc.) but few people then thought carefully about what those different role subsets would want to achieve in their interaction with the team project wikis. The result was a lot of rather bland stories about administrators creating wikis, with rather fewer stories that got to the heart of what the wikis were for: managing communication amongst team members on the project.

Some candidates wrote good stories, but which did not refine the actual function given in the story, or were not close spin-offs of the original function. For example, some candidates wrote stories about lecturers uploading lecture notes for students to see. This doesn't seem to me to be very closely related to the specific topic of "team project wikis".

D) This question aimed to test candidates ability to focus in on areas of major business value to the developer. There were a couple of good answers, but many others were confusing and vague. I had real difficulty in working out just what functionality was being proposed by some answers. Some candidates had difficulty separating out value for the developers from value for the customers. For example, prototyping difficult parts of the system (such as the GUI) is a useful learning tool for the developer, but it doesn't necessarily deliver as much value to the customer as a straightforward but immediately usable bit of functionality.

A handful of candidates misread the question and assumed that the bullet point list of infrastructure components was in fact a list of required systems to be built. This naturally made it difficult to give a coherent answer to the question.

Question 4(SME)

a)

i) This was intended to be a very easy question, to get candidates off to a good start. But in
fact I was able to award the full mark to surprisingly few answers. The main problem was the very ambiguous language used to answer the question. The correct answer was “as soon as the story is written”, but many people wrote answers such as “tests can be written from the beginning” (the beginning of what? The project? The iteration? The universe?) or “tests can be written before any code has been written” (try, but unspecific? The question asked for the "earliest" time at which they could be written). Some answers were even as terse and as unclear as “Now!” and “Immediately”.

A lot of people wanted to write and test-first and test-driven development, though the relevance wasn't clear given that the question was asking about the earliest possible time at which tests could be written - which is earlier than the time when we are even thinking about writing code.

ii) Another question that I had thought was straightforward, but for which few good answers were supplied. Again, the main problem was with ambiguous language: tests, it was claim, can be executed "from the beginning", "once test is completed", "immediately". Some candidates said "test can be executed when they are written", without making it clear whether this meant written on paper or implemented as executable tests. Again, there were many irrelevant references to test-driven development.

iii) In contrast, most candidates answered this question well. There were some perfect answers, and many near misses. The most common form of near miss was to give a perfect set of test columns, except that the column stating whether the customer was a member of the loyalty scheme or not was missing. This resulted in the loss of just 1/2 a mark.

The next most common near miss was not to give quite enough information about the contents of the basket: specifically, to give a column for number of items in the basket, and the cost of only the cheapest of these. This information is enough for us to express the test, but it is not expressed in terms likely to map well onto the underlying implementation or the business model in the customer's heads. We would not really want to store just this information about a basket; we would have details of the specific items ordered and their individual costs. So, the test should really be couched in this more domain-realistic way. Answers which gave a more concise summary of basket contents typically resulted in the loss of just 1 mark, as other aspects of the test table design were typically correct.

Some candidates added an "outcome" or "pass" column, for which I could see no purpose. Some didn't give a "total basket cost" or other column from which the result of the discount could be deduced, but instead gave boolean columns titled "Shipping cost applied?" or "3 items or more discount applied". This is incorrect for the same reason that having a column called "number of items in basket" is incorrect.

I did not penalise students who gave additional (unnecessary) columns, such as customer membership number. Only missing information resulted in lost marks.

Almost everyone correctly identified the ambiguity inherent in the story descriptions. I was mainly looking for the ambiguity regarding the behaviour when both forms of discount applied, but I also awarded a half mark for remarks about the ambiguity of the shipping costs.

iv) There were many excellent answers given to this question. I tried not to penalise candidates again here for errors in the answer to part iii), although this was not always possible. The most common cause of lost marks was a failure to provide actual values to be used in the test, rather than rough descriptions or ranges of values to choose from (e.g., saying ">30" instead of "31").

Several candidates wrote the best answer to this question as part of their answer to part (iii) and then were confused as to what to write in answer to (iv). Similarly, some candidates used better test table columns in their answer to part (iv) than they had given in part (iii). In each case, I marked the answer that resulted in the maximum marks, and ignored whether it had been given under the correct question part number or not.

Very few people gave test cases to disambiguate the required behaviour of the system around the boundary value of 30. (i.e., was the discount applied for orders of 30 pounds or of 31 pounds?)
v) There were some reasonable answers, but a significant number of candidates struggled with this question. Even when reasonable Junit code was given, the designs chosen for the underlying objects were not very sensible, resulting in the loss of up to 2 marks.

The most common major cause of lost marks was when candidates gave fragments of production code instead of test code. For example, code that calculates whether the discounts are applicable to a generic basket is production code, code that checks whether the total cost of a basket is the expected value (i.e. has had the correct discounts applied) is test code.

B)i) Some good answers were given, but many candidates waffled vaguely around the topic, without really answering the specific question that was asked. Many students talked about the generic benefits of the lego-defect-tracking idea, without actually mentioning benefits relating to the BVC aspects of the idea. A number of people complained that modelling the severity of the defects would be highly subjective using this approach, although I am not sure that using numeric scores is all that more objective, is it? Others compared the idea with other agile practices, not mentioned in the question, such as whole-team responsibility and paper prototyping. In fact, a small number of candidates seemed to be mixing BVC up with paper prototyping, saying that it had to be conducted around a single table, and other confusing comments.

ii) Most candidates seemed to be running out of steam by the time they reached this question, and there were no really strong answers. The most frequent causes of lost marks were vague, incomprehensible and highly incomplete answers.

---

**Comments:**

This exam has a very high average (I haven't calculated it, but it will be high). Not entirely clear to me why this should be. I didn't think it looked all that easy when I set it, and the external examiner thought it looked OK, but on the other hand I haven't marked it very leniently—pretty well every mark I've given corresponds to something in the model answer, so I had to give them. Maybe they were just good students (self-selected from people who survived the second year Symbolic AI course as I gave it). The only structural change in the exam was Q1, which was six short questions rather than one long general one, which does have a very high average (it's out of 30, almost no one got less than 20), but I actually thought this would be harder than my usual long general Q1. So perhaps they were just good students—if the average does raise eyebrows then I'd like to see a scatter graph of these students marks on this one vs their marks on other things.
Section A

Q1: This question was very well done. A few candidates were confused that there were no non-zero entries in the table for part (b).

Q2: Despite the clues to its solution being in the gentle lead up to the final part, most who attempted this were thrown by the mention of Tarjan's Algorithm. Most candidates chose to answer Q1 instead.

Section B

Q3. This question was generally very well done, with most candidates demonstrating understanding of the flow optimization procedure explained in lectures and its application to matching.

Q4. This was not done particularly well. Students struggled in particular with the last part, which should have been straightforward for those able to follow the relevant lecture. Candidates who clearly stated that a proposition letter p should be made true if p (regarded as a literal) is reachable from u_0 got the marks for part (i); and candidates who clearly indicated that truth is preserved along directed paths in the graph got the marks for (ii). I wasn't very strict.

Q5. No serious attempts were made on this question. Evidently, students lacked the confidence to tackle it, so this part of the course needs strengthening. This is a shame, because there were many almost free marks for those who had the courage to tackle it.

Q6. The was done moderately well. Using Savitch's theorem to answer part c) was regarded as using a large bomb to kill a small baby: only partial marks were given. Most students struggled with part e), which is horrendous when you consider how standard this is. Again, the lectures should be improved in this area.

Comments:

Please see David Lester for individual feedback.

Overall this exam was very well done, seven out of 61 being 100%.

Q1 Those that remembered how to match the compiler/abstract machine to the natural semantics scored near 20. A number of people didn't bother with part (c) (which was easy), and more than a few missed some of the details in part (a). Everyone attempted this question.

Q2 All but one candidate attempted this question. In the main well done.

Q3 Those with the courage to take this on will have found that it was relatively short to write the answers.

Q4 The most popular third question; part (c) caused more trouble than I would have thought it should with many leaving it blank.

Q5 As usual -- and despite it being even easier than normal -- this was attempted by just a few students. There was one (short) perfect answer!
Comments:

Q1.

46% of the class answered this question.
Average mark: 56%
Std dev: 4.2

The question was generally answered ok. However, the topics and concepts under discussion were not particularly difficult, so I would have expected to see a better quality of answers. The quality of the diagrams was very variable, and in some cases extremely poor, which led to lost marks. About one-third of those who answered seemed to be unaware of the papers on “Zippering” and “Volumetric Diffusion”, despite the need to read them being repeatedly stressed. Since the concepts were also discussed in the lectures, I wonder if these lectures were not attended.

Q2.

54% of the class answered this question.
Average mark: 50%
Std dev: 3.7

It’s probably fair to say that Q2 was slightly more difficult than Q1, but not by much, but the quality of answers left something to be desired. I got the impression that understanding of the various concepts were not "joined up" into a coherent story. Almost nobody was able to sensibly discuss that a calibrated video sequence can be augmented with synthetic CG *because* it is calibrated, and so the OpenGL camera can be set to "that calibration" and when the graphics are drawn, they will be projected correctly into each frame of the sequence. The quality of the supporting diagrams was variable, and was, alas, only within the range of "adequate to poor”.

Q3. 63% of the class answered this question
Average mark: 59%

This question was generally well answered. The quality of the diagrams was very variable, including some that were so small as to be unintelligible scribbles. There was some confusion between the ‘gridcell’ approach and the ‘octree’ approach; though in most cases error-transfer was applied so not all marks for that section were lost. In many cases careless marks were lost by not applying the technique sensibly in the context of the game (and simply saying "in the context of the game" in front of a sentence which then was just a generic description of the algorithm isn't sufficient!)

Q4. 37% of the class answered this question
Average mark: 56%

Generally well-answered, though the descriptions of ray-tracing and radiosity were often on the thin side, and rarely were any sensible conclusions drawn about their applicability in the context of a real-time visualisation. Again, illustrations were often too small and scribbled to be able to attract marks.
The paper was well answered. The optional questions attracted about the same number of takers, and the distributions of marks were similar.

Of the 59 registered students:

2 were absent, but both had never submitted any coursework.
6 got less than 40%
7 got between 40 and 50%
The rest, 44, got 50% or more. The highest mark was 56/60.

As in the past, the main weakness was lack of precision in describing concepts. This is a problem with a high proportion of students – perhaps more than half. Parts of questions involving the reconstruction of a program were generally better answered. This does demonstrate knowledge as reconstructing a program does involve problem solving reflection, and this type of question is as discriminatory as other types.

Comments:

Question 1: It looks like this question was not well prepared by the students. Many students could not answer 1(a), and many missed out the fact that tunnelmode can protect traffic flow confidentiality.

Question 2: Students did this question mostly well, but some of the students were not clear about the difference between DSA and RSA.

Question 3: The most common mistakes were in (b) and (c). In (b), some students didn’t know the Needham-Schroeder protocol. Some students were not clear that the DH algorithm is a public-key algorithm and used for session (symmetric) key establishment. In (c) some students gave two variants of the same session key distribution protocol, one using timestamps and the other using nonce, rather than two different session key establishment protocols.

Question 4: The students did this question pretty well.

Some useful statistics:

In total, 54 students sit in the exam.

Question 1: taken by 24 students, and the average mark is 49%;
Question 2: taken by 54 students, and the average mark is 68%;
Question 3: taken by 51 students, and the average mark is 52%;
Question 4: taken by 34 students, and the average mark is 60%;
The overall average mark = 57%.

6 (out of 54 students) fail to pass the 40% mark; 30 (i.e. 55%) students get results >= 60%.