Most of the exam is MCQs.

There is personalized feedback in ABC.

See the separate pdf for Uli Sattler's feedback.

Here is general feedback for Bijan Parsia's essay questions:

The general schema was 1 point for coherence/intelligibility, 2 points for basic facts, and 2 points for strength of argument.

2.14:

Facts:
Many people did not mention what the problem was that XML tries to solve, to wit, that it attempts to be a (universal) "external representation". External representations have the requirements of roundtripping and self-describingness. I was generally generous with missing the external representation part. 1 point for each requirement (i.e., that it is, what it is, and how XML breaks it).

Few people discussed whether the problem was hard. While XML fails to solve the problem exactly, it's still possible to argue that it's failure is minor (i.e., union types are obscure; leading zeros are pointless). You could lose a point for failing to consider the opposite side (i.e., attack a defence, defend an attack).

In general, it's not really enough to say "external validation destroys self describingness" rather than "external validation destroys self describingness because some of the information needed to convert the external representation (the XML document) to an internal representation (the desired PSVI) is outside the document (e.g., in a Schema), thus the "description" is not inherent in the document."

Some off topic responses:
Error handling (esp. draconian)
No mention of Postel's law was correct
"representing semi-structured data"
communication between people

Most 0s were completely off topic and lacking in coherence. I counted failure to make *any* sort of case for a proposition as a coherence failure.
Most 1s were completely off topic, but reasonably coherent. If you made a sufficient (off topic) case that wasn't utterly off base you could get a 2.

Examples of utterly off base: Labeling as a "problem" the fact that XML doesn't address all use cases; saying that datatypes are a problem that XML solves.

Note that self-describingness does not mean "human intuitable". In this context, it just means we can get to the right internal representation based on the external representation alone.

2.15

A key point to discuss is the "degree" of change between versions. Should we have namespace changes for every version, however small?

It's comparatively rare that different versions of the same element need to appear in the same document. Of course, "when" that happens, different namespaces are likely necessary for disambiguation.
Distinct namespaces per version is completely distinct from contained namespaces. There's no connection.

As emphasised in class, deception, confusingness, etc. are features of namespace *declarations*, not of namespaces themselves. There's no necessary connection between having more namespaces even in a document and being more prone to these problems.

Putting version information per se in the namespace isn't a significant advantage. I.e., if you forced them to be random strings most of the benefits (and drawbacks) of the policy would remain.

It's important to note that both documents *and applications* (can) break and need updating with new namespaces.

Change in namespace doesn't necessary make the difference between versions (or degree of difference) clear absent some policy of "how much" change makes a version. It does let us know which version the document was encoded in. (But one could use a version id attribute.)

---

**COMP 60421** Ontology Engineering for the Semantic Web
Sean Bechhofer
Bijan Parsia

**Comments:**
Please see PDF.

---

**COMP 60611** Fundamentals of Parallel and Distributed Systems
John Gurd
Graham Riley

**Comments:**
The questions in order of popularity were as follows: Q4 (22 answers); Q2 (13); Q1 (12); Q3 (5). The average examination mark was around 53% (compared with 75% for the laboratory-based coursework).

By far the most common reason for low marks was the complete absence of an answer to a part of a question. This might have been because it was taking longer than expected for answers to be written, leaving students with a lack of time to answer questions fully, but this was not obviously the reason in a significant number of cases.

Another cause for low marks was poor answers to parts that could have provided relatively easy marks. ‘Bookwork’ parts that should be easy to answer in an open book examination were one example; the ‘essay’ style part c) in Q1 was another.

In some cases, marks were lost because the answer provided did not address the question that was asked.

The most common mistake for Q1 was that students were unable to generate a correct state space diagram from the simple algorithm given.

The most common mistake for Q2 was to omit an answer (or provide a very poor answer) for part a). Several candidates had severe difficulties in providing an appropriate model for either or both of the partitions in part b).

The most common mistake for Q3 was an inability to generate relevant fragments of a state space diagram from the given algorithm (and, hence, an inability to articulate clear answers). In contrast, part a) of Q3 was answered very well by most students who attempted it.

The marks for Q4 were highly bimodal. 9 candidates got 13/20 marks or above, while 9 candidates got 8/20 marks or below, leaving only 4 candidates near the average. Arguably the most common mistake for Q4 was an inability to provide a cogent answer to part a).
The questions in order of popularity were as follows: Q3 (14 answers); Q2 (13); Q1 (5); Q4 (2). The average examination mark was around 50% (compared with 74% for the laboratory-based coursework).

By far the most common reason for low marks was the complete absence of an answer to a part of a question. This might have been because it was taking longer than expected for answers to be written, leaving students with a lack of time to answer questions fully, but this was not obviously the reason in a significant number of cases.

Another cause for low marks was poor answers to parts that could have provided relatively easy marks. The most obvious examples of this were parts asking about ‘bookwork’, which should be easy to answer in an open book examination.

In some cases, marks were lost because the answer provided did not address the question that was asked.

There were no common mistakes in the five answers for Q1.

The most common mistake for Q2 was an inability to quantify accurately the different sources of overhead in the given example for part b).

Q3 was the most popular question, yet it had the lowest average mark. No candidate achieved a first class mark (14/20 or above) for their answer. Mostly this was because only limited parts were attempted. Apart from this, the most common mistake for Q3 was to fail to realise that the second loop in part b) has a well-balanced workload. Many candidates stated that this loop is triangular, but it is actually a combination of two triangular parts which together always have the same size, thereby equalising the work per iteration of the outer loop. The problem here is actually the access to array a, which is transposed in the second of the two inner loops.

Q4 was answered well by both students who attempted it.
Section A - SS

The questions in Section A required essay-like answers involving a number of topics discussed during the lectures, and represent a balanced blend of book knowledge, application of technique with discussion and original thought questions.

A large number of students attempted to answer original thought questions by copying fractions of text from books, slides and articles, and therefore were not able to get full marks. The students that showed deep understanding of the concepts involved in the questions and were able to describe and apply them in the context of the questions were better rewarded in terms of marks.

Section B - JAK

Summary: most students answered Q1 and generally higher marks were obtained for Q1. Most of the directly taught material seemed to be well answered material indicated as self-study, further reading or introduced in the coursework seemed to be less well answered. A weakness common to many poorer answers was that they failed to make clear assumptions, show working or justify an answer.

Q1
a) Data Warehousing
i. Very well answered on the whole
ii. Some good answers but some quite poor ones
iii. Very well answered on the whole

b) Two classifiers
Overall very well answered in terms of calculations; justification of best classifier was relatively weak.

i) New classifier
Some very good answers, but some quite poor ones.

c) Generally most answers quite interesting and in some cases very good; examples were often not given despite being asked for.

Q2
a) k-nearest neighbours. Disappointing answers in general; area closely related to part of the coursework;

b) Contingency table. Generally well answered in terms of calculation.

c) Information gain. Almost a binary split in answers: very good or quite poor. The poorer answers tended not to make their working clear.

D) Decision tree induction/associative classification. Generally disappointing answers overall; area had been highlighted for self-study.

E) Noise/outliers. Some very impressive answers which focused very clearly on the question and provided good justification; poorer answers seemed to not use the question as the driver and lacked clear justification.
Overall the exam average was 69%, so many people did very well.

A common confusion was on question 6. The key point that many missed was the last sentence "Remember, the probability of rain in general is not the same as the probability of you seeing the rain.", indicating that $p(\text{rain})$ is not the 3/7 suggested by the sentence "You know that in general, in the UK, it rains about 3 days per week". Instead $p(\text{rain})$ must be computed by marginalising over the two classes.

Another common mistake was in q7, where many thought that $p(x)$ could be computed by $p(x_1)p(x_2)p(x_3)$. This is not true. The naive bayes assumption only says the features are "class" conditionally independent, not fully independent. So it is true that:

$$p(x|y) = p(x_1|y)p(x_2|y)p(x_3|y),$$

but not the more general

$$p(x) = p(x_1)p(x_2)p(x_3)$$
Question 1
Average: 75.45
Answered by: 100%
Generally the question was answered well.
(a) some students give wrong definitions/explanations of well-founded sets.
(b) was generally answered well, some miscalculations in sub-question (iii)
(c) was poorly answered. F and G are equivalent if (F<->G) is valid and therefore in order to apply the resolution calculus we 1) reformulate validity (F<->G) as unsatisfiability of ~(F<->G) and then 2) apply clausification algorithm to ~(F<->G) to obtain a set of clauses to which we can apply the resolution calculus.
(d) and (e) were answered well;

Question 2
Average: 66.25%
Answered by: 36.36%
(a) many struggled to pinpoint main differences between structural and syntactic CNF transformation:
(b) syntactic CNF transformation preserves equivalence of formulas whereas structural preserves only equi-satisfiability.
(c) structural CNF transformation is polynomial whereas syntactic is exponential in general.
(d) was generally answered well, many forgot that unit propagation decides the Horn fragment
(e) poorly answered, students have difficulty in formalising properties.

Question 3
Average: 67.7%
Answered by: 100%
(a) i. Generally satisfactory to good explanations. In a few cases examples were given that are not legal first-order formulae. In some cases formulae of second-order logic were given as examples; these are wrong as quantification over predicate/propositional symbols is not allowed of first-order logic.
ii. Few correct answers. Many didn't give an answer or gave a wrong example.
(b) i. Generally answered well.
ii. Answers were mixed. Common mistakes:
- claiming a clause is false in I_C when in fact is true, because it contains a negative literal ‘not A’ and A not in I_C. In that case the clause is true in I_C.
- for productive clauses (i.e. Delta_C is non-empty) to state that C in true in I_C.
- to not list the clauses from smallest to largest but vice versa (not so common)
- to repeat the model construction for the clauses extended with factors and resolvents (not common)
iii. Most answers identified the tautologous clause as redundant. Few got that clause 2 is redundant because it is implied by clause 1 and larger than it.
Question 4
Average: 74.3%
Answered by: 63.6%

(a) i. & ii. Posed no problem
   iii. Was more difficult but most answers correct
   iv. Very few correct answers; most common answer I = \{ p(a), p(f(a)) \} is not correct

(b) Generally answered well. Marks deducted for not using the rules as defined in class (and definition sheet). Occasional mistake: to overlook the side conditions in applying the substitution and orientation rule.

(c) i. Generally answered well. For clause 2, just saying that Q(y,b) contains more symbols than P(y) is not enough. Similarly for clause 3.
   ii. Generally answered well. A few did not use ordered resolution.

COMP 61221  Mobile Systems  Steve Furber

Comments:
Q1: General ARM & RISC architecture question, with an exercise on conditional vs conventional code efficiencies.
   Most students attempted this question, with generally good marks. It seems that they do have a working understanding of the ARM instruction set architecture and the role of conditional execution in that.

Q2: Thumb code question, with a brief write-up of the process of coursework Project B.
   All but one student attempted this question, with generally reasonable results, though they were surprisingly poor at giving a systematic account of how they went about Project B (or they failed to spot that that was what the question was asking for!).

Q3: Low-power cache memories.
   Just over half the students attempted this question. Results were mixed, with few capturing the use of A[1:0] only for byte and half-word accesses. This question required quite a bit of careful block-diagram sketching.

Q4: Memory management.
   Half the students attempted this question, with generally poor results. They didn't catch the distinction between the "principles of operation" and the low-level mechanisms of a memory management system, the former expecting brief mention of virtual memory, relocatable pages, etc, and the latter being best addressed with a block-diagram sketch (which was explicitly asked for). Many answers offered a reasonable sketch and then just repeated the information in the sketch in their text.

Q5: Embedded ICE and on-chip buses.
   A minority of students attempted this question, with very poor results (though some of the students who did attempt it were those who did poorly overall). Not a single student spotted that part c) asked about AHB, rather than ASB/APB which have featured in previous exam questions, so they all completely missed the point of this part of the question.

COMP 61411  Cryptography  Richard Banach

Comments:
Q1: Reasonably well answered.
Q2: Very well answered.
Q3: Very well answered.
Q4: Few answers, mostly dreadful.
Comments:

Question 1:
This question asks for challenge-response protocols, but some students give authentication protocols that are not of challenge-response type. Another common mistake is that some of the designs are not secure (have security loopholes) and/or are not clearly explained.

Question 2:
Students largely did well in this question. The most common mistake is in (a) – not able to illustrate what an ACL (Access Control List) is.

Question 3:
Many students miss out the security requirements of message authenticity, i.e. in addition to message integrity, both parties should ensure that they are talking to the claimed entities. Some use a public key to encrypt the large data file, which is clearly not appropriate.

Question 4:
The students did this question pretty well.

Some useful statistics:
---------------
In total, 60 students sit in the exam.

Question 1: taken by 38 students, and the average mark is 47%;
Question 2: taken by 50 students, and the average mark is 69%;
Question 3: taken by 46 students, and the average mark is 58%;
Question 4: taken by 46 students, and the average mark is 71%;
The overall average mark = 61%.
34 (out of 60) students get results >= 60%.
Section A

Question A1

a). 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]

This was easy and many people got full marks. However, since it was easy, I marked it strictly - you needed to say two clearly separate, relevant things to get the marks. Some people said more or less the same thing two or three times.

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 (product owner) working full-time with the team - the UP does not mandate this. [1]

A lot of people said that the interaction involves different artefacts, which is a valid alternative to the second point above. The most common error was not fully appreciating the role of the stakeholders in the Agile UP - they are involved at the beginning of every iteration, give feedback at the end, and can be consulted as necessary in between.

c) State four different groups of stakeholders, other than students, in the MELT project, and what their principal concerns will be.(4 marks)

For example:
- University Management will be concerned that the software is delivered on time and will provide value for money.
- The Exams office will be concerned that tests will fit in with their existing procedures (The tests are done under exam conditions).
- The English Language centre will be concerned that the software will enable them to test students effectively.
- Technical support will be concerned about how easily the system can be deployed and managed.

Many answers were too vague and/or merely stated obvious functional requirements, e.g. “Question setters will be able to set questions” This kind of thing is very important in real life - if stakeholders’ concerns are not addressed adequately, all sorts of problems occur.

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

They both primarily 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 are normally kept short (and decomposed as required). 1 User stores are often written on physical cards, UCs are normally written electronically. 1

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

This was answered poorly, only two people got full marks. Most problems were to do with misunderstanding of use cases. UCs are text, not diagrams; they can be written at different levels of granularity, they don’t have to be longer than user stories (as the example in the
next part shows); they can be prioritised (remember in the UP we tackle high risk
requirements first); they can be, and are, written and revised throughout the process. This is
understandable to some extent, as relatively little time was devoted to UCs in the course, but
this makes it rather surprising that this was the most popular question in section A. What
happens next is predictable…

e). The following is a possible version of the Take Test use case for MELT.

“The student types his/her username and password into a grey dialog box and presses the
Start button to start the test. The student clicks on a radio button to indicate the correct
answer for each question. A timer stops the test when the time is up.”

Give eight reasons why this is not a good use case description (8 marks)

Note: the students are expected to know the requirements for the MELT case study, as these
have been discussed at length in the lectures. The question is stated in this form rather than
asking the students to write a UC, as experience shows that marking UCs objectively is
difficult and time consuming.

“The student types his/her username and password into a JDialog [user interface specific 1.
technical language 1] and presses the Start button to start the test. [User interface specific,
1. Missing step - rubric is shown first, 1. tests are run under exam condition, so student can’t
start immediately 1] The student clicks on a radio button to indicate the correct answer for
each question. [User interface specific 1.Only applies to MCQs, MELT has other question
types 1] A timer stops the test when the time is up.” [As described, there is no interaction
with the user, so this is a non-functional requirement.1. Omits to say the student can elect to
finish early. 1. Omits to say answers are saved 1]

… nobody got full marks for this part. There were a couple of things people spotted which I
didn’t. The timer is stopped but not started and “correct answer” is rather optimistic!
However: many answers demanded details which would be totally inappropriate at this level
of description (e.g. the format of the username and password); many answers claimed that
the UC is too big and should be split up - as it stands it passes the Boss test, broken up into
Login etc. it would not.

Another, more subtle very common error was saying that the UC should specify how the
student should navigate the test; this would require UI details. (The UC could specify that the
student can answer the questions in any order and come back to them, but that’s a level
above the details of navigation. In ABC this is always true while there are several modes of
navigation specified at exam setup time )

Question A2

a). How can UML diagrams be used to facilitate communication between software
developers in an agile development team? Your answer should mention at least two different
uses.
(4 marks)

All UML diagrams are ceremony, and hence optional - it would be un-agile to spend a large
amount of time on diagramming. [1]. Domain class diagrams can be used to help understand
the domain, particularly for developers new to the project [1] Design class diagrams can be
used to discuss design choices [1] UML diagrams can be used at the start of a sprint to help
determine architectures choices for that sprint. [1]

Marks for anything else sensible, e.g. they can be used at any point in a sprint to clarify the
team’s understanding of the domain, design choices etc.

The original wording of this question allowed any use of UML diagrams within Agile
development, but on review this was considered too easy. This more specific question, about
“communication between software developers” threw a lot of people. Many answers would
have been fine with the original question but talked about stakeholder interaction which is
out of scope for this question. Many people failed to say four different things.
b). Explain, using an example from MELT, 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.[1] The diagram can then be inspected to identify area of uncertainty and generate further questions for the stakeholder (multiplicities are a particularly good source of questions [1]). An example from MELT is that there is a requirement for two or more invigilators per room, not per test (any other valid example is fine). [1] For stakeholders with an appropriate technical background, it may be most efficient to show them the diagram with areas of interest marked [1] Otherwise the diagram should not be show, but is list of questions arising from it should be prepared in advance. [1]

Most answers failed to make the last point above (which I discussed at some length in the lecture). Otherwise answers were ok except that many of them failed to make fives separate points.

b). What is the fundamental relationship between a domain class diagram and a design class diagram? (1 mark)

Domain classes often inspire design classes, although this is not a 1-1 mapping. (No marks for answering any other questions about domain classes and design classes, e.g. replacing “relationship” with “difference.”

A few people made the mistake predicted in the sentence above, but most got it right.

c). State three things which are shown on design class diagrams which do not appear on domain class diagrams. (3 marks)


Well, answered in general, a fair number of people lost a mark for missing out pure fabrications, which as whole extra classes are the biggest addition. A few people said that inheritance relationships are added - if there is a natural inheritance relationship between domain classes (e.g. Question and subclasses) it should be shown on the domain class diagram.

d). The Irwell Media Store started off as a bookstore, but later started selling CDs and DVDs too. More recently it has started selling laptops too, and it plans to branch out into other electronic devices such as phones and PDAs, although it does not intend to become a general store. The company is taking its first tentative steps into the online marketplace, and has commissioned you to produce a prototype inventory system

Draw a domain class diagram which represents the kinds of products which the store currently sells, in a way which can be easily extended in the future. Hint: the best solution has 7 classes of which 3 are abstract. 5 marks

The ideal solution is a pure inheritance hierarchy with an abstract Product class at the top, and abstract subclasses Meida and Device (or similar). The Media class has concrete subclasses Book, CD and DVD, while the Device class currently only has Laptop as subclass, but the description indicates that others such as PDA may be added in the future.

Numerous errors were made on this simple task:
- Failure to read the question correctly: only current products are required, not future ones or the store itself.
- Failure to use UML notation correctly. The only relevant connector is the one for inheritance (fatarrow). Most people lost marks for this reason alone.
- Failure to use the mapmaker principle: names which don’t appear in the description. In particular, Media is the best name for the superclass of Book/CD/DVD (or you could treat Book separately and have a DigitalMedia class.
- Use of plural names for classes - class names are almost always singular nouns: the
description says “books” but the class name should be Book.
-Addition of extra gunk not mentioned in the description.

e). If you were modelling a general store with thousands of different kinds of products, how
would the domain model have to be different? 2 marks.

It would not be feasible to have a class for each different kind of product. There would either
be a single Product class with some sort of product description, or Product could have a
small number of subclasses representing different broad categories of product.

This is tricky and most people didn’t get it. You can’t change the code every time a new kind
of product is added (except perhaps by some cunning code generation scheme, but nobody
suggested that). Well done to those who spotted the problem.

Question A3

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]. The form the “building blocks” for design patterns. [1]

Answers with examples are also good.

Answers to this were very disappointing; none made either the second or third point above.

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

There were some very good answers to this question, and a few very poor ones, which
showed no understanding of wither Polymorphism or PV. In between, common reasons for
losing marks were failing to use examples from MELT and failing to appreciate that PV is
something you aim to achieve, rather than something (like Polymorphism) which you make
use of.

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]

A lot of people explained indirection and gave good examples, but failed to relate it to
polymorphism or PV. A lot of people got fixated on Controllers, which are indeed an example
of indirection, but not a particularly good one in this case because they’re not obviously
related to Polymorphism. Also, several people failed to point out that a Controller provides
PV by allowing the model and the view to vary independently.

d) In ABC 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 modes, 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.

Answers to this were generally good but most were not complete and accurate enough to get full marks, e.g. if you elect for a Use Case controller it’s a good idea to say what the use case is (e.g. Invigilate Exam).

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

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

Maybe people were running out of time, because the answers to this were very sketchy. Just naming GRASP principles (and some people failed to do even that!) is not enough as the question relates specifically to the monitoring tool.

Section B

Note: question 1 is about Agile processes in general and invites the students to write “mini essays”. Question 2 is more about technical details.

Question B1

a) In Agile software development individuals and interactions are valued more than processes and tools. State three other key principles.(3 marks)

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

Working software over comprehensive documentation (or minimise ceremony)
Customer collaboration over contract negotiation
Responding to change over following a plan

Although most people got full marks, a significant number could not reproduce the above or equivalent, and a couple answered another question entirely!

b) In agile approaches individuals and interactions are valued more than processes and tools. Explain how this principle is reflected in agile practices (5 marks)

There are many agile practices which reflect this, for example continuous. interaction with a customer representative is considered essential. [1] Within the development team, interaction is encouraged through stand up meetings and retrospectives. [1] Pair programming, done properly, is a highly interactive activity. [1] Processes and tools are secondary but that does not mean that we do not value them too [1]. For example in a TDD/BDD model of development the only way that the development process can proceed is because tools make automated testing possible [1].

Marks for other sensible points, but for full marks something along the lines of the last two points is necessary.
Very few people made five distinct points at all, and fewer still recognised that processes and tools are valuable (“it’s not that we don’t value the things on the right…”). Also, several people waffled about how interaction is a Good Thing in general rather than citing specific agile practices.

c). You are tasked with introduction of agile methods into a global fleet vehicle hire organisation that produces computer software for its internal needs. What arguments would you use to persuade senior and middle management that this is a good move for the organisation? (4 marks)

The challenge here is to justify Agile in non-technical terms. Some good points to make are: Agile is now mainstream, having been around for more than a decade; 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. In short, Agile is a very cost-effective way of producing software.

Few people made four points relevant to the situation. A number of answers included technical terms which would be inappropriate for this part of the question. A number of answers made what could have been good points but were expressed in an unclear manner.

d). In the same situation as in part c, what additional arguments would you use to persuade technical managers and staff? (4 marks)

Here we need to refer to specific Agile practices. For example: Agile builds on established iterative development practices, which are generally agreed to be more effective than older methods; test-driven development ensures full test coverage and improves software quality, as does automated regression testing. Pair programming provides continuous code review, improves design quality and ensures that the “bus number” is at least 2. Overall the focus is on producing code rather than ceremony.

Answers to this part were generally better, although the points raised were generally not as distinct from those in the previous part than they would need to be in practice. A number of people mentioned staff motivation, which would have been better as part of the answer to the previous part. Again, a lot of people failed to make four distinct points.

e). “Agile is more than just a development process, it is also an attitude or mindset” Explain this statement, and state whether you agree with it, and why. (4 marks)

Many possible answers but, for example: Timeboxing can be applied in many situations, and arguable helps with time management in general; the idea of “retrospection” and self-improvement can also be widely applied; likewise the emphasis on physicality and body language. I expect positive answers to the last part, although my personal experience of agile practitioners suggests otherwise.

Indeed there was 100% agreement with the statement. Again though, many people failed to make four separate points, and a number of people answered the question in terms of personal benefits of doing agile development as opposed to “life in general”. One particular answer I though was especially apt. The second paragraph reads:

“I personally agree with this statement, because it is a healthier way of living, many people prefer to be alone using the computers instead of socializing and meeting new people, or trying to help other people, particularly in university or school environments it is very common that the most capable students don’t help the ones that have difficulties. This attitude would lead us to a better world.”

Question B2

a). State four important activities which take place at specific times in a typical Scrum sprint. (4 marks)
Sprint planning meeting at the beginning of the sprint [1]; Daily stand-up at the start of each day [1]; Customer Demo at the end of each sprint [1]; Retrospective at the end of each sprint [1].

This was very well answered. A few people spent a lot of time telling me lots of details of e.g. retrospectives rather than finding four different things.

b) How does tool support enable an agile process? (3 marks)

Automatic testing tools enable TDD [1] and by building up a regression test suite supports refactoring to deal with change [1]. Version control tools are essential and support a fast develop-build-deploy cycle. [1]

This proved to be very difficult to mark because the question should, with hindsight, have explicitly said "software tools" and almost everybody interpreted it more widely. The criterion I tried to apply was that a tool is something that pre-exists (such as a story board) rather than an artefact you produce (such as a user story) but this distinction is not always as clear-cut as in that example.

c). Explain the notion of velocity in agile processes and state two different ways in which measuring velocity can help to maximise the amount of useful work during a sprint. (3 marks)

Velocity is the number of story points that a team can implement in a given time. [1]
Measuring velocity enables us to add or subtract user stores during a sprint [1] and helps to improve estimates of velocity for future sprints [1]

Most answers were good. Some people missed the point of dynamic adjustment during the sprint. A few answers said things like knowing your velocity is too low makes you work harder, which is not the point.

d). Draw a burndown chart for a team which has underestimated its velocity (4 marks)

X-axis is time [1] Y-axis is story points remaining [1] estimated velocity is straight line \ [1]
actual velocity is below the estimate (so team would finish early).

I was rather picky marking this, so I deducted a mark if it wasn’t obvious which was the reference line and which was the actual line, and also if the actual line ended at the bottom, since that would only be the case if the team’s actual velocity was double that expected (in which case it would have been corrected before then. I didn’t mind whether the correction was shown or not provided it was shown correctly. I awarded one mark for a correctly drawn chart for a team which had overestimated.

e). What action should the team take in the situation described in part d? (2 marks)

In collaboration with the customer representative [1] the team should add extra story points to be implemented during the sprint [1]

Several people got it the wrong way round. Many failed to mention the role of the customer representative. I gave 1 mark for other sensible ways of using the extra time, e.g. improving code quality.

f). The networking infrastructure in your office is causing problems, although there is some disagreement within the team as to how serious the issue is. How would you go about resolving this in a retrospective? (4 marks)

Firstly the team needs to agree that this is a genuine impediment [1]; the facilitator should ensure that everybody gets a say, and that a more serious issue is not being overlooked [1].
Assuming the team does agree on this, an experiment needs to be designed for the next iteration. [1]. For example someone with networking expertise might be delegation to investigate the network traffic with a view to improving the routing [1] (Any other reasonable experiment is ok).

An issue which arose when marking this is that about a third of answers interpreted “you” as
“me personally” rather than “my team”. The latter was intended although the question should have been explicit about this. The problem with “me personally” answers was that they tended to fixate on how to convince the team there was a problem, missing other important parts of the process. Many answers failed to suggest an experiment specific to the problem (so were more or less completely generic. Others fixated on the disagreement within the team (which is hardly likely to be a huge issue) and ignored other aspects. Otherwise, answers were generally good.
Comments:

Q1: Most answers correctly identified the advantages of compositing. Marks we're lost in describing the mask in a 2.5D rendering system - it does not specify transparency but occlusion. The proof that the 'Over' operator is not commutative was well done. Part b) on RenderMan was where marks tended to be lost. Separating the modelling and rendering domains allows rendering technology to advance (usually more quickly). A range of compatible renderers can also be used. Other features of RenderMan were concerned with efficient memory usage (due to the streaming nature of RIB files) allowing very large scenes to be rendered. Provision of shading language, in particular displacement mapping, allows rich surface detail and high quality images. RenderMan on the GPU requires mapping of various RenderMan shader types to GPU shader types.

Q2: All candidates attempted this question. Most people knew the quaternion formula and how to apply it to a point to rotate it. A detail was that the point should be represented as a pure quaternion. Euler angles were usually adequately described although few mentioned that they decompose a rotation into independent rotations (about the principle axes). Marks were lost in describing the user interface used for quaternions (use Euler angles in the UI, operations performed in quaternion space) although the description of how to convert Eulers to Quaternions was accepted and usually correct. The description of why to interpolate from q1 to q2 or -q2 was a little vague (shorter path in quaternion space resulting in less twisting/rotating of the object through intermediate rotations) but most knew the method of choosing between q2 and -q2, using the quaternion dot product. Marks were often lost on the final question - safe to convert Quaternions to matrices at rendering time because no further rotation operations are performed on the matrix.

Q3: All candidates attempted this question. Most knew the difference between FK and IK. The justifications for which to use for the dinosaur animation were often a little vague. Ideally use IK for short leg chains, FK for long tail chain to produce appearance of muscle-driven motion (which IK tends not to produce). Not all could provide the description of the IK algorithm but those that did usually gave a good answer. A detail often missing was that the algorithm generates joint angle velocities and so a final integration step is required to produce joint angles. The descriptions of parametric interpolation were OK, often given by an example of a motion graph. Wanted details about functions of an independent variable. The arc length parameterisation question accepted answers describing the small-stepping method in 3D path animation as a solution as well as the Newton-Raphson method for motion graphs. The description of the problem should have mentioned that equal changes in parameter do not necessarily produce steps of equal length along the curve.

Q4: Where sections were answered the answers were usually reasonably complete. Marks were lost where entire sections were not answered. Time may have been a factor with this question. The main problem with control hairs is the speed of computation (O(N^2)) when dealing with collisions. Occlusion culling to speed rendering should be considered. A description of the four hair modelling techniques was book work. The key-hairs question related model vertices to key-hairs. Collision and control only applied to key hairs, not intermediate generated hairs. The control vertices referred to the CVs on the key-hairs, not the location at which key-hairs are placed on the model. The main problem with beards is that the underlying model deforms during lip-syncing and beards cast more shadows on to neck geometry.

Q5: A new question this year hence not very popular. Time may have also been a factor if answered at the end of the exam. Some details missing in early parts regarding marker extraction. Some parts were not answered, particularly on constructing motions from a database, where details about locating samples in the database, continuity of motion and
coordinate invariance had to be considered. A little more detail was needed to describe how to correct the problems seen with motion capture, although the problems were correctly identified.