BCS HIGHER EDUCATION QUALIFICATIONS
BCS level 5 Diploma in IT

April 2010

EXAMINERS' REPORT

Software Engineering

General Comments

This is a technical paper about Software Engineering. Questions seek to test candidates' knowledge, and ability to apply that knowledge. It is a poor answer that simply describes recalled information. It is a good answer that can then show application of the recalled knowledge.

This exam encourages reflection, meaning critical review of what the topic means in the wider context of software engineering. Candidates are encouraged to read more widely about 'high end' goals of software engineering, such as risk management and ethics, and reflect on how development processes hinder or help the higher aims.

Overall, the candidates' responses to the Software Engineering examination questions showed a good understanding of the subject. Most marks for individual questions are in the average range.

The rate of response was unevenly distributed for booklet A: questions q1, q2 and q3 received roughly the same attempts (some 50-60% of candidates). Most candidates instead considered questions Q4 and Q6 in booklet B. Considering the questions, this is understandable: q6 required to provide justification for the Graphical Interfaces within CASE tools; while q4 required the student to identify and define different testing techniques and terminology. Quite unsurprisingly, question Q3, regarding UML diagrams, was very popular among candidates.

Question 1 – Software Processes

a) Give TWO examples of how conflicting requirements can arise among the stakeholders of a software project during the elicitation phase, and, for each, plan resolution actions.

b) After taking part in a series of successful software projects, you have been recently appointed a project manager within a software engineering organisation. Your first project in this position will be to build a large web-based application. The team has experience of doing this type of work before, and in all of the previous projects, requirements have been thoroughly documented by the customers.

The above scenario is an outline specification for a software project

1. What team structure would you choose and why?

2. What software process model would you choose and why?

Answer Pointers

a) A good answer addresses these points:

It includes any two examples illustrating the different requirements between two or many different stakeholders. Large commercial-off-the-shelves systems with thousands of customers are more prone to conflicting requirements. Examples about conflicting requirements between final customers, software architects, developers or even managers can be considered valid.
The resolution would happen in stages:

a) Isolating one or several “viewpoints” (inter-actor, indirect or domain viewpoints) would be the first part of the analysis;
b) identifying the conflicting requirements, done also in collaboration with the stakeholders and by tool-based analysis
c) generating resolutions, manually by stakeholders, and by tool-based analysis
d) selecting the relevant resolution, manually by stakeholders

b) A good answer addresses these points:

Although the client has trusted and domain-specific relationships with customers, this project is more about infrastructure development using existing processes. As such, there are few unknowns and those that exist can be readily defined with some systems analysis. The candidate in summary should realise that the presented case study is a straightforward project. (5 points)

In terms of the **team structure**, a closed paradigm team structure is one option. Since requirements are well defined, it will be possible to partition requirements and allocation to sub-teams. The large size of the project also mitigates in favor of a Controlled decentralized team (communication among individuals and subgroups is horizontal, vertical communication along the control hierarchy also occurs).

In terms of the **software process model**, since there is no discussion of schedule, we assume that delivery date is reasonable. Therefore, it might be possible to use a linear sequential process model (work has been done before). However, an iterative model (e.g., spiral) is also a good possibility. 5 marks

Examiner's Guidance Notes:

Q1.

Part (a) was answered extremely well in very few cases, while in general students failed to identify simple examples where stakeholders could face conflicts or diverging views. From the examples provided by a handful of candidates, it emerged that the term "stakeholder" could apply to several actors within the same software development team.

Part (b) was more evenly distributed, but the question about the "team structures" probably the weakest part for the majority of students attempting this sub-question. In general, there was a very weak understanding of what different team structures are and what advantages could each bring in terms of a software project.

Question 2

Project Management

a) Consider the following scenario:
You have been asked to build a software system to support a web-site for real-time video-streaming. The system will need to serve a minimum and a maximum number of on-line users, with a given threshold of reliability; also, since it is used on the network, it will need to implement appropriate measures to counter-act to attacks against the system.
List and describe at least THREE possible technology risks that one would face in a project of this type?

(15 marks)

b) Suggest TWO techniques for estimating the effort and the costs associated to this project.

(10 marks)

Answer Pointers

A good answer addresses these points.

a) a) The goal of the risk mitigation, monitoring and management plan (RMMM) is to identify as many potential risks as possible.
Phase of Mitigation—how can we avoid the risk?
Phase of Monitoring—what factors can we track that will enable us to determine if the risk is becoming more or less likely?
Phase of Management—what contingency plans do we have if the risk becomes a reality?

b) If a risk is high probability, high impact, but the project team cannot do anything to mitigate, monitor, or manage it; it should not appear in the risk table. For example, if the company goes out of business in the middle of the project, the impact is catastrophic. Assume the probability of this is high, given outside business pressures. The software project team will likely have no control over the outcome.

c) Some of the associated technological risks are
r1 – The web underlying programming languages are obsolete in a time-span that is shorter than traditional programming languages, and which results in lack of support and professional coverage;
r2 – real-time streaming requires appropriate hardware that has to be updated regularly for making front to increasingly large audio/video formats ;
r3 – bandwidth is required for larger files to be streamlined;
r4 – stolen copyrighted material is included,

Examiner's Guidance Notes:

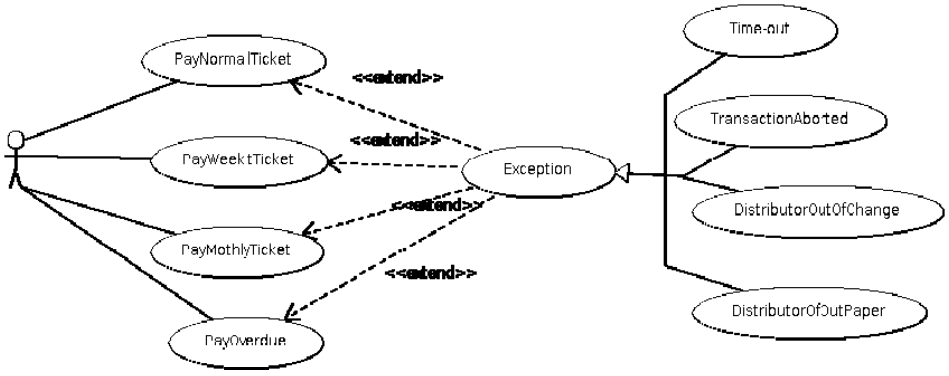
Q2.

For Part (a), this question asked candidates to identify risks related to a given scenario. Most of the students attempting the question could produce a set of adequate risks, and they clearly identified the main issue of the same scenario, namely the fact that the depicted company works in a volatile environment, and where the main constraints are based on hardware resources.

For Part (b), candidates generally used generic techniques as Delphi, CUSUM or COCOMO. While not wrong per se as answers, the candidates in general did not justify properly why one technique would give an advantage over others in the effort estimation of such project, which should be mostly based on estimating hardware resources.

Question – OO Design

Consider the following UML Use Case scenario:



a) list and comment the Use Cases associated with this scenario

(10 marks)

b) describe the use of the <<*extends*>> and the <<include>> notations

(10 marks)

c) draw an additional Use Case, in order to update the tariff of the tickets.

(5 marks)

Answer Pointers

A good answer addresses these points.

a) The following are the use cases of the diagram

PayNormalTicket: for the customer to obtain a single ticket (train, metro, parking ticket)

PayWeeklyTicket: for the user to have a seasonal (weekly) ticket

PayMonthlyTicket: for the user to have a seasonal (monthly) ticket

PayOverdue: for the user to be reminded to pay the overdue ticket

UpdateTariff: a different actor should be identified for this use case

The following exceptional cases have also been included:

Time-out (i.e. the driver took too long to insert the right amount),

TransactionAborted (i.e. traveller selected the cancel button without completing the transaction)

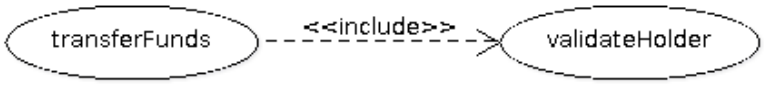
DistributorOutOfChange

DistributorOutOfPaper

b) The <<extends>> use case consists of one or several behavior sequences (segments) that describe additional behavior that can incrementally augment the behavior of the base use case. Each segment can be inserted into the base use case at a different point, called an extension point.

You can add extend relationships to a model to show the following situations:

* A part of a use case that is optional system behaviour
* A subflow is executed only under certain conditions (i.e. exceptions)
* A set of behaviour segments that may be inserted in a base use case



The <<include>> relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case)

- The behaviour of the inclusion use case is common to two or more use cases.
- The result of the behaviour that the inclusion use case specifies, not the behaviour itself, is important to the base use case.

c) the following use case should be displayed, making sure to have another actor, the System (not the user) which is updating the tariff.

Examiner's Guidance Notes:

Q3.
Part (a): most of the students attempting this question could identify the various use cases. Not all the candidates recognised the Exception as a generic use case, or the use of the Design Pattern

In part (b), also most of the students gave proper descriptions and examples for both the extends and the include keywords of UML.

Part c) Given the answers of part a) and b), it was quite surprising to realise that very few students could draw the additional case study, and properly realise that customers of these use cases cannot update the tariff as they want, but another actor (the "system") has to do it.

Question 4 – Testing

a) Distinguish between the following: Unit testing, Integration testing and User Acceptance testing

(15 marks)

b) Describe the use of “test drivers” and “test stubs” in the phase of Integration Testing. Explain how they are used in the top-down and bottom-up approaches to integration testing, and provide an example for each approach.

(10 marks)

Answer Pointers

A good answer addresses these points:

a) **Unit testing** is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation, and affects

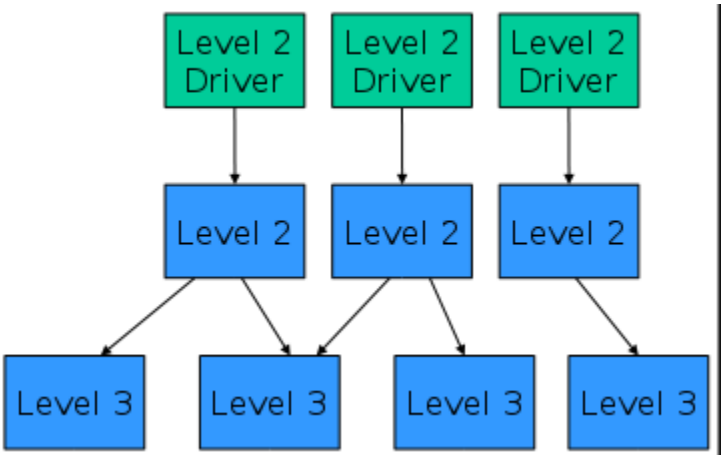
- Testing all operations associated with an object;
- Setting and interrogating all object attributes;
- Exercising the object in all possible states.

Component testing is a software development process where groups of related classes are tested together. This can be achieved by a white-box component testing, or a black box one.

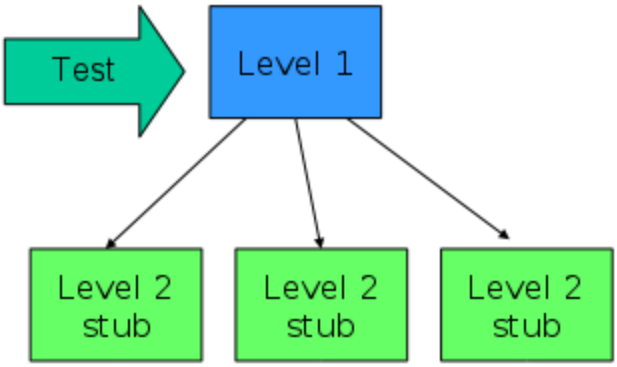
Integration testing is a software development process in which program units are combined and tested as groups in multiple ways. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution.

b) The bottom-up method and the top-down method are the two major ways of carrying out an integration test.

Bottom-up integration testing begins with unit testing, followed by tests of of progressively higher-level combinations of units called modules or builds.



In top-down integration testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems and networks.



Examiner's Guidance Notes:
Q4. Generally students were able to give good answers to all parts of this question. In particular, the candidates attempting the question could properly reproduce the visualization of a top-down and the bottom-up approach to integration testing.

Question 5 – Development methods

a) Consider a word processor and outline THREE possible types of errors that can occur relating these to ways the software implementing the word processing could fail, by completing the following table in your answer book:

Type of error	Example of this error type in word processing software

(9 marks)

b) For two of the examples given in part a) provide a suitable error message that would be appropriate to inform users in the event of failure occurring and explain the reasons why this can be considered informative to the users.

(10 marks)

c) Discuss the difference between a software error, a program fault and a system failure illustrating your answers with relevant examples.

(6 marks)

Answer Pointers

A good answer addresses these points:

a.

Type of error	Example of this error type in word processing software
System error	The word processor's internal buffer overflows and cannot save the user's text during a large insertion.
User error	The user attempts to use a font that is not on the system, e.g. mistypes Arial as Ariel.
External error	Due to the user's lack of access rights, the word processor is unable to save the file produced when the user requests it to save.

b. Case 2: The font you have chosen, Ariel, is unavailable on this system. Please check that you have spelt the font name correctly. Do you want to use it anyway? Yes/No

This lets the user know that their choice is unavailable on their system. It indicates that this may in fact be a typing error. It gives them the option of continuing with their original choice which may be available on other systems.

Case 3: An internal buffer overflow has occurred. Only the first 256 characters of your insertion have been saved. You will have to make another insertion to achieve your goal of inserting over 256 characters.
This explains what has happened and it also offers the users a workaround the present limitation on the buffer size.

c. **Software error** A software system state that can lead to system behaviour that is unexpected by its users. For example, the user expects the word processor to save all the text they have entered but the software erroneously truncates long lines of text.

Program fault A characteristic of a computer program that can lead to a software error, for example, a failure to initialise a variable which could lead to the wrong value being used.

System failure This is an event that occurs at some point in time when the system does not deliver a service expected by its users. A web site crashes when its users place a heavy demand on its services and cannot respond in the time expected.

Examiner's Guidance Notes:
Q5. Very few students attempted this question, and several could not give good examples to locate user or system errors. There was also a general lack of understanding of the differences between an error and a fault.

Question 6 – CASE Tools

a. Explain what is meant by CASE tools in the context of software development and distinguish between upper and lower CASE tools giving examples of each.

(5 marks)

b. Discuss the role that CASE tools play throughout the software life cycle phases and identify specific tools that a software development team would typically use to support their activities.

(15 marks)

c. Many CASE tools employ a software repository. Outline the role of the repository and discuss how it could be used to in developing a programme of software reuse.

(5 marks)

Answer Pointers

A good answer addresses these points:

a. CASE stands for Computer Aided (or Assisted) Software Engineering and CASE tools are software systems used by software developers to aid/assist them in their work of developing software and managing software development projects. Upper CASE tools are used in the strategic planning and management of the project; they focus on the upstream activities at the start of the project, e.g. project planning software or requirements management software. Lower CASE tools are used in the down stream activities and support the technical work of engineers designing, developing, and testing the software, e.g. a UML editor, a compiler.

b. Usually CASE tools are used throughout the software life cycle starting with upper case tools to do the project planning and resource management, track issues that arise. During the requirements phase, a requirements management tool may be used and also prototyping tools may be employed to explore various options and user interface possibilities with users. CASE tools that support UML may be used at this stage to diagram the Use Cases and then later on to design the Classes and more detailed aspects of the design. Some UML Case Tools will generate code stubs. Other CASE tools support code generation for reports generation and user interface design. During the implementation phase, compilers and interpreters will be used to translate the software written in high level languages into an executable form. During the testing phase, a test oracle can be used. There are various test automation tools that support automatic running of test cases and recording of results. Finally during the software

maintenance phase, tools to support reverse and re-engineering can be used, e.g. tools for code analysis and impact analysis. Throughout software development, various CASE tools to collect metrics can be used.

c. The repository is a central store where the associated intermediate and final products of software development can be stored in a controlled and versioned form. It provides a basis for collaborative working amongst the software development team and managers can view its contents to monitor progress of the team. Where a repository has been in use across several projects, common software products can be shared and this can provide the initial basis of a software reuse programme.

Examiner's Guidance Notes:

Q6. This question was by far the most chosen by candidates and the one that achieved the best results among all. Most students were able to list the benefits of software reuse and describe its practice but not explicitly with reference to reuse within the software life cycle. Also, most students could distinguish between Upper CASE and Lower CASE tools, and identify which tool could be used and in what phase of the software life cycle.