THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2007

EXAMINERS' REPORT

Software Engineering 1

General Comments
This is a technical paper about Software Engineering. Questions seek to test candidates' knowledge, and their 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.
Question 1 was not well done, though popular. Most candidates understood the direct result of design methodologies, to partition a complex issue into many smaller issues; however, the reflective parts were not well addressed.
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 ethics and people management, and reflect on how development processes hinder or help the higher aims.
Responsibility towards customers cropped up in several places in this exam, with a worryingly wide range of views. The ethical and quality norms of Software Engineering are plain; customers rarely know what they want, but it is our professional duty to help them towards an understanding of what we can do for what cost. Customers are not infallible, neither is their ignorance to be exploited. It is our professional duty to maintain continuous skill development to stay abreast of the rapid changes in our technology.
Some candidates had not completed the front cover with the number of the question and part-questions attempted. This made it very difficult to find all the parts of an answer, then determine whether it was a continuation of a previous answer, and assess and record the mark for that answer as a whole.
Nonetheless, the candidates' responses to the Software Engineering examination questions showed a good understanding of the subject. The full range of marks demonstrates this and most marks for individual questions are average or above. A good spread of attainment was evident.

Question 1

a) Discuss how a design method encourages teamwork by breaking down a problem into many smaller parts. Illustrate your answer by describing how a design method assists reviews, interoperability and maintenance. You may choose any design method known to you, and discuss how reviews, interoperability and maintenance are aided by your method. **(12 marks)**

b) Outline an ethical approach to communication within a software engineering project team. **(5 marks)**

1

c) A recurring problem in teamwork is variable, individual productivity. Discuss how you would manage this variation for the benefit of the whole team. **(8 marks)**

Answer Pointers

This question was designed to test candidates' knowledge of how design processes work in a craft business like software development, and to promote reflection on the integration of design with two major concerns in Software Engineering – ethical behaviour and variations of individual productivity. Many candidates did not produce a defensible conclusion for part (c); answers were inconclusive, suggesting that the issue of varying productivity has not been much considered during preparation for this exam.
A good answer addresses these points.
a) Any design method is acceptable if the candidate shows how it breaks the problem into parts, often with some checklist to guide the breakdown and limit its extent.

Reviews: the breakdown makes a problem tractable, designed in small and well-defined units. This makes review of internal intention (algorithm or data) and external interface (API) precise and exact.

Interoperability; the breakdown makes for well-defined interfaces, the easier to make interoperable.

Maintenance: modularity with low coupling is well-known for ease of maintenance. A decent design will have structured the development into its parts with, at worst, reasonable coupling so making the maintenance job that much easier.

b) The answer should show a clear understanding of 'ethics' – it can be expressed as 'ends and means' meaning targets and processes, or results and pressures on people to produce them. Or it could be the observation that there is always tension in a team when targets seem hard or frustrating to reach. Or it could mention bullying in the workplace, the 'blame' culture.

An ethical approach could be informal-personal, about trust in a manager, or keeping faith with promises made. Or it could be formal, such as codes of conduct for tolerance or treatment of all grades of personnel.

c) Team work has a range of roles and needs a range of skills for delivering a software product. The Productivity thing usually means 'coding' but could also mean 'timekeeping'.

Strategies to combat slow coding could include reassignment to testing, or even some skills appraisal to find the strengths of the individual – for communication, documentation, testing or customer relations. Find the area of an individual's best contribution, and promote it.

Examiner's Guidance Notes

Candidates answering this question showed a good awareness

Question 2

Suggest an appropriate generic software process model or models that might be used as a basis for managing the development of the following systems:

a) an e-auction system which will run over the web,

b) a modification to a computer game,

c) a system to control radiation therapy administered to patients in a hospital,

d) a new virtual learning environment for a college or school, and

e) an interactive web-based system that allows customers to review films available and book tickets for a cinema.

In each case, justify your choice by giving reasons that take into account the type of system that is to be developed. **(25 marks)**

Answer Pointers

This question was designed to test candidates' skill to select an appropriate development model for each type of scenario described. Scenarios are often used in this way, to encourage choosing and selection. Of course, choices must be defended with reasons. Absence of selection, or reasons, indicate lack of knowledge about how specific problems in this vocational business of software engineering are addressed by specific methodologies.
A good answer addresses these points:

1) **An e-auction system which will run over the web.** Depending on the customer requirements, a component based software engineering process could be used if preliminary searches reveal that suitable reusable components are available; if not, then the spiral model could be employed as the requirements are prototyped and better understood leading to ultimate use of the waterfall model in the system development.

2) **A modification to the computer game.** In this case, the most appropriate model would be a combination of the component-based software engineering model and software maintenance/evolution model as this will involve a modification to part of an existing system, and probably involve the reuse of an existing game framework, a new component, the modification will need to be developed and integrated to create the new system.

3) **A system to control radiation therapy administered to patients in a hospital.** Given the safety and life critical nature of this system, waterfall model using formal methods is appropriate as this will enable software developers to ensure that the software has been correctly derived from the initial specification through correctness preserving transformations, the properties of the initial formal specification also can be subject to formal proof. The initial requirements and specification will of course have to be validated with all the domain experts: the radiologists, the doctors, technicians, equipment manufacturers, etc. Rigorous testing will still be required.

4) A new virtual learning environment for a college or school.
Here a development process that allows for a high level of interaction with the end-user base would be appropriate so an incremental development process would be appropriate, such as that found in the Extreme Programming or Agile development processes.

5) **An interactive web-based system that allows customers to review films available and book tickets for a cinema.**
The component-based software engineering process with a high degree of software component reuse could be applied here as there are a number of COTS suitable for achieving such a system. Some rapid prototyping may be desirable when developing the user interface and here representatives of the intended customer base would need to be used.

Examiner's Guidance Notes

Candidates answering this question showed a good awareness of various software process models and their applicable to different types of system development. There were some generic answers, but most candidates took note of the salient features of the systems and suggested appropriate models.

Question 3

a) Explain what CASE is in the context of Software Engineering. **(5 marks)**

b) Give examples of two CASE tools that are relevant to software project managers. **(4 marks)**

c) Give examples of two CASE tools that are relevant to software developers during the design, implementation, or testing phases. **(4 marks)**

d) In the case of each of the tools described in your answers to b) and c) above, classify the tool according to:

- its main functions, and
- the phase(s) of the software life and particular activities where it is applicable. **(12 marks)**

Answer Pointers

This question is designed to test candidates' knowledge of automated tool support for the development process, and the different types of toll support available.
A good answer addresses these points:

a) An explanation of CASE in the context of Software Engineering.
CASE, Computer- Aided Software Engineering, refers to the software used to support the software life cycle and its various activities, such as planning the software project, requirements engineering, system design, coding, testing, reuse, maintenance, and other more general activities such as documentation, change management, risk management, timesheet processing, etc. CASE tools are specific tools used and they may be part of a Software Engineering Environment such as those supplied with Eclipse or Rational.

b) Examples of 2 CASE tools that are relevant to software project managers.
Any of the following: PERT tools, estimation tools, e.g. COCOMO based tools, spreadsheets, Microsoft Project, Software Configuration management tools, e.g. SourceSafe, CVS, Subversion; etc

c) Examples of 2 CASE tools that are relevant to software developers during the design, implementation, or testing phases.

Any of the following: editors, especially programming language specific editing systems, prototyping tools, e.g. user interface generators, design tools, e.g. Argo UML, Rational Rose; Compilers; Interactive debugging tools, etc.

d) In the case of each tool described above, classify the tool according to the parameters:

Tool	Functions	Phases	Activities
Editor	Input and manipulate text, including source code, and diagrams	Throughout, every phase	Documentation throughout and specialist editors during design and coding
User interface generator	Generates user interfaces through GUI where developer can drag and drop predefined components of the UI such as buttons, dropdown tables, etc.	Requirements and Design and Implementation	Used in initial prototyping during requirements specification working with customer/users and in the development of UI component of the final system.
SCM	Controls versions of the systems and its components as well as associated documentation	Throughout the life cycle	In all activities where code and documents are being changed and changes must be made in a controlled fashion.
Estimation tools	Allows manager to estimate the cost of system development	Used prior to start of development and at key points of risk assessment if employing the Spiral model.	Estimation is used in initial planning and is particularly relevant during any major project re-planning.

Examiner’s Guidance Notes

The progressive nature of this question allowed candidates to develop their exposition on CASE, give specific examples, and further to classify these in terms of functionality and applicability in the lifecycle and to support specific activities. Most candidates attempting this question gave well reasoned answers.

Question 4

a) Many writers assert that quality processes during development reduce the cost of software maintenance. Give your reasons for accepting, rejecting or modifying this view. **(10 marks)**

b) Discuss how Software Quality Assurance interacts with the people who perform formal technical reviews, configuration management, and testing. **(15 marks)**

Answer Pointers

This question was designed to test candidates' knowledge about the application of quality assurance principles, not just their description of definition.

A good answer addressed these points:

a) A brief discussion of the cost areas during maintenance, and making a relationship between them to the cost of QA during the development process, to establish where costs are incurred, and to derive support for the view postulated by the question.

The cost of maintenance is the costs of operation, repair, replacement, or product evolution/product novelty.

Requirements discipline, design reviews, testing, and validation all contribute to development QA and to reliable operation and repair/replace during maintenance.

An interesting area is 'how much is enough?', leading to a discussion of residual error rates and market comparisons with competitors' products to ensure both that you are slightly better and also that you don't spend too much on QA to be needlessly superior, to meet market expectation. Contrariwise arguments will be taken on merit.

b) SQA and its interaction with formal technical reviews: creates an audit trail, gathers metrics, acts as a training session for new staff, promotes use of standards (design and code).

SQA and its interaction with configuration management: identification of parts, version control for product reliability, configuration test and regression test for customer satisfaction, reporting for SQA audit trail.

SQA and its interaction with design: Design is a creative process, and not amenable to semantic test though a design method can be compliance-tested. The way to test the capability of a design to deliver the intended outcome is to compare with similar designs and similar products.

Examiner's Guidance Notes

Too many candidates avoided 'taking a side', thereby evincing ambiguity about where and why quality principles fit in software engineering. Other answers reckoned that any and all changes asked for by customers had to be developed for free because 'the customer is always right', and some answers thought that the responsibility of developers stops after handover and that someone else pays the maintenance bill. A principle of quality standards in software is that customers rarely if ever know what they want, hence the popularity of prototyping and incremental development in all their forms. And a principle of accounting is 'nothing is free'.

.

Question 5

a) Explain how the different types of testing are related to the phases of the Software Development Life Cycle. **(10 marks)**

b) Discuss the role that software documentation plays throughout the software life cycle and its importance in supporting both development and testing. **(8 marks)**

c) “Software testing can only detect errors present in a software system; it is not possible to show through testing that a system is 100% error free”.

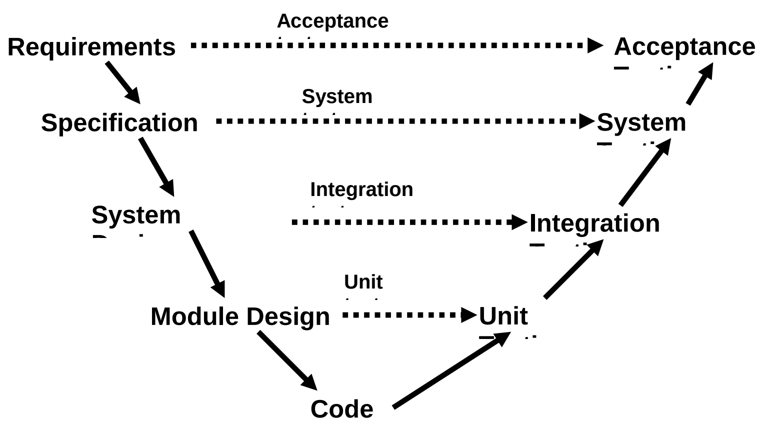
Discuss the validity of this statement with respect to modern software engineering testing practices. **(7 marks)**

Answer Pointers

This question was designed to test candidates’ knowledge and application of testing across the lifecycle, and sought some critical commentary on effectiveness of testing on quality.
A good answer addressed these points:

a)

SLC and Testing with linking SLC



Or similar diagram or an account in words covering the above.

b) Software documentation has a very important role in supporting communication amongst all who are involved in the software life cycle. It is used in communicating to the customer. It is used to communicate between the project manager and the software development team. It is used to communicate between team member and particularly when new people join the project. Early documents produced during the development are used in the planning of testing and in the development of specific tests, and are important during maintenance to aid system understanding. Clear documentation of test is required so that regression testing can be carried out during maintenance.

c) There are various reasons why testing cannot eliminate 100% of errors in any large scale software development. During testing, one error can mask or hide others. Once an error is discovered, it will be impossible to determine if other anomalies are due to a new error or are side effects of the first error. While some types of programming errors such as syntax errors, variables assigned to but never used, can be detected using static analysis tools, tests which dynamically exercise the program code only demonstrate its behaviour under the specific test conditions. It is possible that testers have overlooked particular test cases.

Testing is also an expensive and laborious phase of the SLC; often, cost trade-offs are made once the system is considered ‘good enough’ for operational use. Some forms of exhaustive testing are impossible within the time limits of the development.
With Cleanroom testing, a more quantitative measure of testing can be determined using statistical measures of software reliability; however, even here, it is not possible to achieve 100% reliability.

Examiner’s Guidance Notes

This question examined candidates' understanding of testing and the links between the Software Life Cycle, its earlier work products and the testing activities downstream. Weak candidates failed to make these links although surprisingly many went on in the second part to mention this partially in their answers concerning the important role of documentation in software development. The final part of the question attracted a range of answers; surprisingly, some candidates were convinced that modern software engineering testing methods can produce error-free software.

Question 6

Consider the following outline specification for a Car Alarm project.

The car alarm system is to be capable of:

- Taking input from an array of sensors that can include switches, pressure sensors and motion detectors
- Operating a siren, able to create a variety of sounds so that a distinctive sound can be picked by the user
- Wireless on/off control, particularly from a key fob
- There will be an auxiliary battery so that the alarm can operate even if the main battery gets disconnected

A microprocessor computer control unit will monitor the sensors and operate the alarm. Your O-O documentation is to guide the developers of the microprocessor code.

a) Identify the objects you would select for a first level of refinement. **(5 marks)**

b) Define an initial selection of two operations that act on the objects identified in a). **(10 marks)**

c) What extra information would you need to develop the system? **(10 marks)**

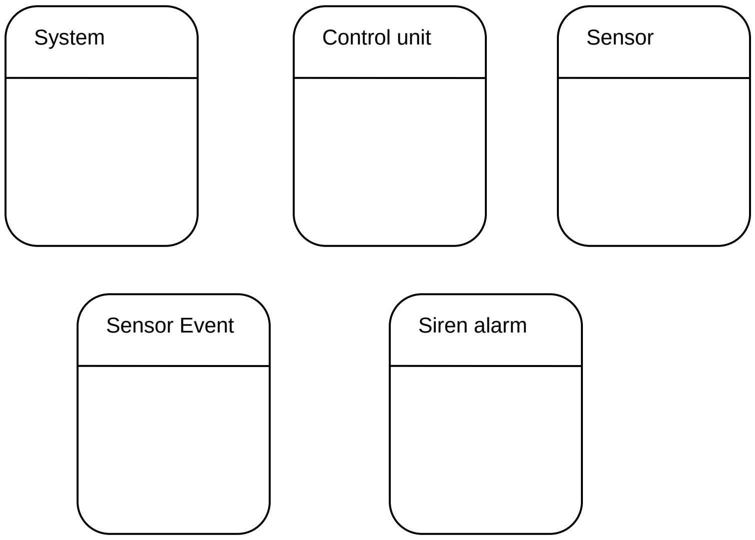
Answer Pointers

The question was designed to test a candidate’s ability to select and model key objects in a fairly complex specification.

A good answer addressed these points:

This question seeks a fairly simple OO model that gets the essentials and recognises unnecessary detail to be omitted.

a) first-level object choices.



Initial objects

b)

Operations: two from

System: on/off, configure audible alarm, log events
Control unit: on/off
Siren: on/of, choose siren audio style
Sensor: on/off, disable/enable.

c) Extra information needed:

- We need to know the control sequence for alternating between ‘on/off’ and ‘select audio preferred’ on the control unit.
- We need to know if siren volume is to be controlled.
- We need to know if there is a siren time-out.
- We need to know more specifics about the number and type of sensors.
- We need to know if there is additional illumination incorporated within the control unit.
- We need to know how the system constructs and maintains its internal log.

Examiner’s Guidance Notes

Some candidates modeled battery switch-over from main to auxiliary as driven by software; this was a piece of question complexity added to act as a distraction. Most electrical switch-over systems in alarms have analogue logic to maintain power from a backup battery; creating a software switch is needless complexity.