BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 6 Professional Graduate Diploma in IT

SOFTWARE ENGINEERING 2

Friday 30th September 2016 - Morning

Answer **any** THREE questions out of FIVE. All questions carry equal marks.
Time: THREE hours

Answer any Section A questions you attempt in Answer Book A
Answer any Section B questions you attempt in Answer Book B

The marks given in brackets are **indicative** of the weight given to each part of the question.

Calculators are **NOT** allowed in this examination.

General Comments

The following maybe significant in explaining the overall low pass rate:

1. Coverage of the syllabus. Candidates used common sense interpretation as a substitute for depth in basic principles and concepts of software engineering rather than foundational knowledge. In terms of breadth of coverage, many candidates appear to have good knowledge of traditional methods, but limited knowledge on modern methods, tools, and techniques as identified in the new syllabus.

   Furthermore, particular areas of the syllabus such as Software metrics (Q1), and on OCL and UML (Q5) exhibited the poorest set of responses from the candidates.

2. Subject awareness. A successful learner should provide more breadth in responses given to all parts of the question by reading more widely publications within the profession as well as recommended text.

3. Examination techniques. There are still instances of candidates answering a question in too much detail resulting in no attempt or very “shallow” responses to other parts of the question itself, the likely result of which is insufficient to compensate for the marks lost by not completing all components of questions.

4. Presentation. It is important that candidate responses to questions are legible, well-structured, formatted and appropriate to the questions and rubric of the paper itself.

SECTION A

A1.

a) Reliability and Usability are important software quality attributes. Give a brief explanation of both attributes and discuss which of these attributes is easier to quantify and measure.

(8 marks)

b) Object Points and Function Points are general, high level system size metrics. Which aspects of the software system are taken into account in the Object Point metric, and which in the Function Point metric?

(8 marks)

c) ISO standards define a set of software quality characteristics and sub-characteristics. They specify which quality characteristics are influenced by which sub-characteristics.

i. Discuss briefly quality sub-characteristics affecting software reliability and software usability.

(5 marks)

ii. Explain why highly reliable systems tend to be less efficient.

(4 marks)

Answer Pointers

A good answer should cover the following points:.

a) There are many definitions of software reliability, but most of them define reliability in terms of probability of failure free operation e.g. “Software reliability is the probability that a system will operate without failure under given conditions for a given time interval”. This implies that reliability can be easily quantified and can be measured directly.

Usability can be defined as “The capability of the software product to be understood, learned, used and attractive to the user when used under specified conditions” or as “The ease of understanding, learning, using, etc.” All software aspects mentioned above are subjective and can’t be measured explicitly. This implies that usability is more difficult to quantify and measure.

b) Object points are based on the following ‘elements’ of the application:
number of screens, number of reports, number of 3GL components .

Function points are based on the following ‘elements’ of the application:
number of input types, number of output types, number of enquiry types, number of logical internal files, number of interfaces.

c)

i. According to the ISO standard (e.g. 9126), reliability is influenced by the following sub-characteristics: maturity, fault tolerance, recoverability, reliability compliance. At least two sub-characteristics should be discussed.

According to the ISO standard, usability is influenced by the following sub-characteristics: understandability, learnability, operability, attractiveness, usability compliance. At least three sub-characteristics should be discussed.

ii. For highly reliable systems (e.g. safety-critical systems) some extra precautions must be taken and implemented. Basically when a fault occurs, the software must continue to operate. This of course requires an extra code. In general, extra often redundant code to detect exceptional conditions reduces software execution speed and increases memory requirements. Therefore, higher reliability results in lower efficiency.

Examiner's Guidance Notes:

This question assesses a candidate's knowledge and awareness of software metrics and software quality characteristics/attributes.

More than 55% of candidates attempted this question, but only a small number produced reasonable answers. In general, there is evidence that the results are poor mainly due to unclear and irrelevant answers.

Many candidates did not provide proper explanations/definitions of software reliability and usability in part (a). Many answers did not answer the question.

Part (b) was reasonably answered by a small number of candidates only. Many answers were irrelevant e.g. object-oriented metrics were discussed instead of object points.

Only a small number of candidates answered part (c)(i) reasonably well. Many answers were irrelevant or/and incorrect. The same comment applies to part c(ii).

A2.

a) With respect to Lehman's laws of software evolution, state the two most fundamental laws and explain their implication for software lifecycle management.

(5 marks)

b) When you are assessing a legacy system, you have to look at it from a business perspective and a technical perspective. From a business perspective, you have to decide whether the business really needs the system. From a technical perspective, you have to assess the quality of the system and its related support software and hardware. You then use a combination of the business value and the system quality to take one of the following informed decisions: scrap the system, re-engineer the system, replace the system, continue the system's maintenance.

Your task is to assess legacy systems in your organization and decide what would be the most appropriate strategy for maintaining these systems.

i. Discuss possible factors you would use when assessing the technical quality of the legacy system.

(10 marks)

ii. Assume that you assessed four systems and the results of the assessment are as follows:

System A: high quality, low business value
System B: high quality, high business value
System C: low quality, low business value
System D: low quality, high business value

What would be your recommendations for each of these systems? Justify your decisions.

(10 marks)

Answer Pointers

A good answer should cover the following points:

a) Law of continuing change: A program that is used in a real-world environment necessarily must change or become progressively less useful in that environment.

Law of increasing complexity: As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying its structure.

The second law states that, as a system is changed its structure is degraded, so its maintenance becomes more difficult. The only way to avoid this is to carry out maintenance in a well organised and systematic way and to invest in preventative maintenance.

b)

i. Possible factors are:

Understandability – How difficult is it to understand the source code?
Documentation – What system documentation is available?
Data – Is there an explicit data model? To what extent is data duplicated? Is the data up-to-date?
Performance – Is the performance of the system adequate?
Programming language – Is the language still used? Are modern compilers for the language available?
Configuration management – Are all versions of all parts of the system managed by a CM system?
Test data – Does test data for the system exist? Is there a record of regression testing available?
Personnel skills – How skilled are people who are involved in maintenance? Are there people available who understand the system?

ii. System A: This system doesn't contribute much to the business, but it may not be very expensive to maintain. It is not worth to replace it, so normal maintenance may be continued so long as no expensive changes are required. If expensive changes become necessary, it should be scrapped.

System B: It has to be kept in operation (because of its business value). Its quality is high, so we do not have to invest in replacement or re-engineering. Normal system maintenance should be continued.

System C: Keeping this system in operation will be expensive and because its business value is low it should be scrapped.

System D: This system is making an important business contribution so it cannot be scrapped. However, its low quality affects its maintenance costs. It should be re-engineered to improve its quality to reduce maintenance costs. Another possibility is to replace the system if a suitable off-the-shelf system is available.

Examiner's Guidance Notes:

This question assesses a candidate's knowledge and awareness of software maintenance and related topics such as legacy systems and software evolution.

This question was attempted by 67% of candidates.

A significant number of candidates did not answer part (a). This topic of software evolution is one of the main topics in software engineering.

Some students provided reasonable answers for part (b)(i), but the evidence shows that many answers did not answer the question.

Part (b)(ii) was answered reasonably well. Many candidates provided proper recommendations for all four systems, but some answers were irrelevant.

B3.

a) Compare and contrast the main features and practices of the agile approach and more traditional approaches at each of the key phases of the software development life cycle.

(16 marks)

b) Discuss how the clearly identifiable good practices in agile methodologies can be effectively incorporated into any software life cycle environment.

(9 marks)

Answer Pointers

A good answer should cover the following points:

a) The traditional software development life cycle of analysis, design, implementation, testing, and maintenance still thrives in many areas of the computer industry today. Sometimes referred to as Plan-driven development, the approach to software engineering is based around separate development stages planned in advance, the outputs to be produced at each stage. In contrast, Agile development inter-leaves specification, design, implementation and testing, where outputs are decided through a process of negotiation during the software development process.

   In the case of agile methods, the focus on the code rather than the design, imply a reversal of the development process, even the issue of testing prior to

implementation. Agile approaches adopt an iterative approach to software development. However, some traditional methods have adopted an iterative approach, but limited to such areas as design, implementation, and testing. Agile methods attempt to deliver working software quickly amidst changing requirements. This is difficult for most traditional methods to accomplish.

b) There are many practices in agile methodologies. Some of these include: active stakeholder participation, collective ownership, display models publicly, prove concepts with code, and use the simplest of tools. Team composition, expertise, and cohesion play an important part in the success of agile methods. Stakeholders in team membership and collective ownership are important outputs of team building exercise. These could be effected during such stages as requirements gathering and specification. However, its focus on small, tightly-integrated teams may result in problems scaling agile methods to much larger systems.

   Subsequent project stages often include elements of plan-driven and agile processes. Deciding on the balance depends on the level of detail in the specification or design before moving to implementation; furthermore, the extent to which an incremental delivery strategy to customers for rapid feedback is realistic. In general, practices where models can be displayed publicly for public comments be it a design document, or prototype application and its code can incorporated into any software lifecycle environment. This allows the opportunity for a greater degree of scrutiny in identifying bugs, ambiguity, or missing requirements.

Examiner's Guidance Notes:

More than 80% of candidates attempted this question, and it had the second highest pass rate. Part (a) was answered reasonably well, demonstrating good knowledge of traditional life cycle methods and some appreciation of the agile philosophy and existing methodologies. In contrast to part a), there is evidence that many candidates had difficulty identifying areas within SDLC where specific agile practices might be employed.

SECTION B

B4.

a) Compare and contrast the following pairs of software lifecycle models, giving particular attention to the application of tools, techniques, and project life cycle phases as progress is made towards a complete system:

i. The V-Model and Evolutionary development

(9 marks)

ii. Extreme programming and Incremental development

(9 marks)

b) Discuss the extent to which the choice of lifecycle models impacts, influences, and determines project test planning and testing techniques.

(7 marks)

Answer Pointers

A good answer should cover the following points:

a)

i. The V model emphasizes the verification and validation of the product. The emphasis is on planning for verification and validation of the product in early stages of its development. The cycle is sequential but with parallel pairing of activities. These include the following pairings:

- Project and Requirements Planning vs Production, operation and maintenance
- Product Requirements and Specification Analysis vs System and acceptance testing
- Architecture or High-Level Design vs Integration and Testing
- Detailed Design vs Unit testing
- Coding

Verification and validation tools are important throughout, but are significant components within requirements specification, design, and implementation CASE tools.

In Evolutionary Development, a prototype is built during the requirements phase. This is subject to end-user evaluation and feedback. Further refinements are made until the end-user is satisfied. It is at this point “real” coding takes place to meet user performance and design and build engineering standards for the final product. The cycle stages could include:

- Develop initial story board
- Discuss with stakeholders to produce partial requirements specification
- Produce preliminary project plan
- Designer builds prototype with key functionality
- Demonstrate and subject the prototype to user evaluation
- Refine prototype until the user is satisfied
- Perform Engineering design and build

Prototyping tools are important throughout the cycle, and form significant element within requirements elicitation, and HCI design and implementation CASE tools.

ii. In XP some phases of the SDLC appear to be bypassed to speed up the development process. Phases have reduced scope and are usually less formal without boundaries,

Coding is the key activity throughout the software project. Using vague or rapidly changing requirements, teams communicate using code, requirements and design behavior of objects are defined early in test cases in the same manner. Therefore, Low-level case tools able to also produce high-level elements (such as designs) through techniques such as reverse engineering and re-engineering are key.

In incremental development, requirements are prioritized and then implemented according to priority. Unlike, XP, distinct phases of design, implementation, and testing are clearly defined with verification being an important element across all these stages. Only a partial implementation of the total system is constructed. Thus, each subsequent release of the system adds functionality, until the total system is built.

High-level yet integrated case tools for generating and prioritizing requirements, designs, coding units are important in the success of incremental development.

b) Lifecycle models generally give emphasis and importance to different aspects of software development. Therefore, models such as XP and V, makes test planning and testing techniques the focus of the development process. In this case, the term test-driven development would aptly describe such methods.

In contrast, evolutionary prototyping may be more concerned about getting the requirements right and the right product will follow. Whilst the Waterfall model emphasis is on fixed stages, budget, and delivery deadlines.

Examiner's Guidance Notes:

This question assesses a candidate's knowledge and general awareness of a range of process models. It was the second most popular question attempted, with the third highest pass mark (39%). There were some very good answers, but there is evidence that most candidates had good knowledge of one of the four process models and spent much time describing models such as V in great detail.

B5.

a) Discuss how the use of Object Constraint Language (OCL) in UML makes designs more logically robust and easy to understand. Illustrate your answers with OCL statements.

(10 marks)

b) Explain what is meant by the static and dynamic semantics of behavioural diagrams. Give some examples.

(5 marks)

c) Discuss whether UML models can become the primary means of communication within development teams and the contract between developers and customers.

(10 marks)

Answer Pointers

A good answer would typically be:

a) The Object Constraint Language (OCL) is part of UML and is a language that allows the visual UML models to be described or annotated with greater precision. Thus, the language specifies such things as constraints on one or more values/components of an object-oriented model.

Thus constraints, makes explicit the parameter, upper and lower limits, and ranges for component behavior. Further, its formal semantics help to reduce or remove any ambiguity in UML models.

As an example, a model of a university admissions system could make explicit that applicants must be at least 18 years old. This can be specified in a UML class diagram as an OCL invariant as follows:

```
Applicant
age >= 18
```

where “Applicant” is the class on which the “age” attribute in every instance of the class must be at least 18 years old.

Pre and post conditions can be used to state explicitly, the assumptions about the pre and post operational state of a class object. For example, the number of applications made can be represented using OCL as follows:

```
ApplicationForm:: completeAppForm()
pre: result   = 0      --  comment no previous application made
post: result = 1      --  Application form completed
```

b) The static semantics represent relatively stable aspects of a system, often structural in nature includes such things as class and object diagrams for visualizing, specifying, constructing, and documenting systems.

The dynamic semantics represent the changing aspects of a system, often time-related. Such diagrams include sequence and activity diagrams.

The use of “tagged values” placed near the UML element as text strings enclosed in braces { }, can modify the semantics of the element to which it relates.

The use of OCL constraints which extends the semantics of a UML element

c) Highlight the increasing adoption of UML in modern developments alongside the growth of the open source and object-oriented movement;

Note the wide range of freely available supporting and integrative case tools, developed to support UML diagrams and associated semantics;

Note that for UML to accomplish this universality of use and application, present opportunities and benefits of using a common language should be identified. This may include greater productivity through automated processes and developer expertise; and, the increased process and product quality through early identification of defects and ambiguity.

The real barriers to the realization of these benefits are: customer education in terms of understanding, using or appreciating concepts (often abstract) of modeling solutions and very formal language supported design of artefacts; the existence and continued maintenance of legacy systems; and the continued drive for specialized tools, and implementations techniques with stringent performance requirements.

Examiner’s Guidance Notes:

This question assesses a candidate’s knowledge and general awareness of OCL and UML development. The question was the least popular of all questions and exhibited the second lowest pass rate (36%). In most of the candidate answers the evidence shows a general lack of subject knowledge with no real grasp of key concepts, principles and practices regarding OCL.