THE BCS PROFESSIONAL EXAMINATION
Professional Graduate Diploma

April 2017

EXAMINERS' REPORT

Software Engineering 2

General Comments

The pass rate was 48% compared to the previous year's results of 50% and 44% for the periods of March and September respectively. The following issues were noteworthy:

1. Coverage of the syllabus. The evidence suggests a lack of knowledge around topics such as Software Process Improvement and Software Management. Centres should be reminded that the entire syllabus should be covered.

   The evidence also suggests that much foundational knowledge was missing, resulting in candidates using common sense interpretation as a substitute for depth in basic principles and concepts of software engineering. For this reason, candidates are encouraged to adopt a staged approached to their learning, by completing the Software Engineering 1 module, before attempting Software Engineering 2.

2. Subject awareness. Candidates should provide more breadth in responses given to all parts of the question by reading more widely and accessing publications within the profession as well as recommended texts.

3. Examination techniques. The evidence suggests instances of candidates answering parts of questions without regard for their indicative weightings, resulting in too detailed an answer for one component part that cannot compensate for the marks lost by incomplete or shallow coverage of the second and subsequent components of questions.

4. Presentation. It is important that candidate responses to questions are legible, well-structured and formatted. Further, if the question asks for a report, the response should be structured as a report.

Section A

Question 1 [Software metrics and models and associated measures of software quality]

a) Write a brief overview of the various forms of software process metrics available today, and discuss how they might be usefully employed from the initial project stages, through to the commissioning of a new system. Illustrate your answers with examples.

(10 marks)

b) Maintainability is one of the most important software quality characteristics. Give the definition of software maintainability.

It has been suggested that maintainability is influenced by the following software quality sub-characteristics: analyzability, changeability, stability, testability. Justify this claim.

(10 marks)

c) Consider the following software attributes: Maintainability, Cyclomatic complexity, Lines of Code count (LOC), Reliability, Number of errors.

Which of these attributes can be measured directly and which indirectly? Justify your answers.

(5 marks)

Answer Pointers

A good answer should cover the following points:

a) A good answer should:

Highlight the importance of high quality process (input) that can result in a high quality product (output). Metrics facilitate prediction, costing, and management decision-making on the process, and expected products;

In requirements analysis and specification, metrics can help to highlight critical aspects of the expected product, its quality and subsequent maintenance, based on available process and resource inputs. For example, a simple function point analysis may highlight the scale of the development and likelihood of delivery within timescale such that the stakeholders are asked to revisit the requirements, re scope and cost the system;

In the design phase process metrics can highlight complexity, productivity, and engineering build quality. For example, the McCabe complexity metric as an indicator of complexity can result in design decisions about the process of decomposition, and subsequent testing and maintenance processes.

In the implementation and testing phase metrics can verify and predict operational performance, configuration and maintenance requirements. For example, a metric for testing in terms of defects identified per 100 modules inspected, may cause

stakeholders to release the product early, in the sure knowledge of the number and timing of patches that would follow in terms of maintenance.

In the maintenance phase, process metrics are concerned with change, their frequency, the sub-systems affected, and the predicted cost and expected system lifetime. Process metrics such as these can lead to decisions to delay certain requests, or system decommissioning.

(10 marks)

b) Maintainability- the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in the environment and in requirements.

(2 marks)

A good answer should provide definitions of analyzability, changeability, stability, and testability. This should be followed by a brief justification for each of these sub-characteristics. For example – testability can be defined as the capability of the software product to enable easy verification/validation. So, if the product has been modified and these modifications are relatively easy to be verified/validated, then its maintenance effort/cost decreases.

(8 marks)

c)

Maintainability –indirectly (explain)
Cyclomatic complexity – directly (explain)
LOC –directly (explain)
Reliability – indirectly (explain)
No of errors –directly (explain)

(5 marks)

Examiner's Guidance Notes:

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

More than 75% of candidates attempted this question, but only some produced sufficient answers. In general, this question caused more problems than questions 2 and 3.

The evidence suggests that only a small number of candidates provided a sufficient brief overview of software process metrics required in part a) and were able to correctly discuss how they might be usefully employed in various project stages. Many candidates discussed software product metrics instead.

Part b) was answered well by a small number of candidates. The evidence suggests that many answers were irrelevant e.g. many candidates discussed maintenance instead of maintainability. Also, only a small number of candidates provided sufficient explanations of quality sub-characteristics and correctly justified their influence on maintainability.

Part c) was answered well, however only a few candidates provided correct justifications of their answers.

Question 2 . [Software life cycle models]

a) What is the main idea behind an incremental development process? Illustrate your answer using, as an example, a system with 15 functional requirements (leading to 15 user services) where requirements Fr1,…Fr5 have the lowest priority (i.e. they are the least important), requirements Fr6,….Fr10 are more important, and requirements Fr11,….Fr15 are the most important to users. Discuss the advantages and disadvantages of incremental development.

(8 marks)

b) Assume that you are a project manager of three projects with the following characteristics:

- Project 1. A complex real-time system whose requirements can be relatively easily identified and are stable.
- Project 2. A web-site for a local library. Requirements are vague and are likely to change in the future.
- Project 3. An order processing system with a web-site for a local business. Requirements are vague but stable (i.e. unlikely to change in the near future).

Consider also the following software development approaches/models:

waterfall, incremental, evolutionary prototyping, throw-away prototyping and component-based development. Which of the above models would you choose for each of your projects? Your choices should be properly justified.

(9 marks)

c) Discuss the view that modern life cycle models with their emphasis on prototyping, create systems that are often fragmented and difficult to integrate; of unsatisfactory reliability, performance, and functionality; and of limited longevity.

(8 marks)

Answer Pointers

A good answer should cover the following points:

a)

Functional requirements are prioritised (divided into different groups with different priorities). A number of delivery increments are then defined (corresponding to different groups). The increments are developed either sequentially or in a phased parallel manner. For example – requirements Fr11-Fr15 are included in the first increment, requirements Fr6-Fr10 in the second and requirements Fr1-Fr5 in the third increment

(4 marks)

2 advantages and 2 disadvantages to be given.

(4 marks)

b)

The following (or suitable alternatives) would be expected:

Project1: Waterfall (No need for prototyping and requirements must be precisely specified)
Project 2: Evolutionary prototyping (requirements are vague and unstable, so there is a need for rapid prototyping leading to the final product)
Project 3: Throw away prototyping 'merged' with waterfall (requirements are vague, so there is a need for prototyping, but they are stable, so the waterfall approach can be used – to develop the back end sub-system in particular)

(3 marks for each)
(9 marks)

c)

A good answer should: Present an overview of the commercial environment within which most software is produced, namely one of: strong competition; available and easily accessible productivity tools; and very knowledgeable and discerning consumers of software.

Argue that, given the changeable nature of customer requirements, prototyping and incremental developments are key approaches by which software developers can reduce the risk of producing the wrong product on time, and the right product in an untimely manner.

Recognise that the shortage of professional developers, and the lack of regulation to the profession of the self-taught developer, can give rise to the problem of poor quality and inefficiency in the use of resources resulting from excessive system fragmentation, poor performance, and concealed defects.

Acknowledge that prototyping will still play an important role in eliciting requirements today and in the future, especially as user needs are often hybrid, and more complex.

Further, the increasing adoption of software openness and reuse may limit the excesses of fragmentation and increase the reliability and longevity of systems.

(8 marks)

Examiner's Guidance Notes:

More than 70% of candidates attempted this question.

Part a) was answered well, but some candidates did not illustrate their answers using the correct example.

Part b) was generally answered well.

Part c) caused some problems. The evidence suggests that a small number of candidates addressed all the issues. Many answers were irrelevant.

Question 3 [Software reuse and component based software development]

a) Explain the difference between software reusability and software reuse.

(4 marks)

b) Discuss briefly benefits of and problems associated with software reuse.

(10 marks)

c) Component based systems development (CBSD) methods place a lot of emphasis on component reuse, hence they differ from 'traditional' systems development methods.

You are asked:

(i) to briefly explain the main differences between 'traditional' and CBSD process/life cycle models;

(4 marks)

(ii) to discuss the main stages of CBSD methods.

(7 marks)

Answer Pointers

A good answer should cover the following points:

a) Software reuse is a software engineering approach where the software development process is geared to reusing existing software.

Reusability is one of software quality characteristics. In general it is the extent to which a program (or its parts) can be reused in other applications. In other words - the ability to be reused in other applications.

(4 marks)

b) A good answer should cover the following points:

Benefits:

Increased dependability – reused software which has been tried and rested in working systems, should be more dependable/trustworthy than new software because its design and implementation faults have already been found and fixed.

Accelerated development – reusing software can speed up system production because both development and validation time should be reduced.

Reduced project risk – the cost of existing software is already known, while the costs of development are always a matter of judgement. So reuse reduces the margin of error in project cost estimation.

Standard compliance – some standards, such as user interface standards, can be implemented as a set of standard reusable components.

(6 marks)

Problems:

Increased maintenance costs – if the source code of a reused software system or component is not available then maintenance costs may be increased because reused elements may become increasingly incompatible with system changes.
Lack of tool support – CASE toolsets may not support development with reuse.
Creating and maintaining a component library – this can be expensive.

(4 marks)

c) The following (or suitable alternatives) would be expected:

(i) The main idea of CBSD approach is the building of systems from already existing components. This assumption has the following consequences for the system life cycle:
The process of component-based system development differs from 'traditional' development processes. The main difference is in the separation of the development process of components from the development process of systems. For the system-level process, the emphasis is on finding the proper components and verifying/evaluating and integrating them. For the component-level process, design for reuse is the main concern.
Component assessment is a new (possibly separate) process for finding and evaluating components.

(4 marks)

(ii) It is expected that the following stages will be briefly discussed:
Component qualification (finding and evaluating)
Component adaptation,
Component composition (integrating)
Component engineering (design for reuse and implementation of 'new' components)

(7 marks)

Examiner's Guidance Notes:

More than 75% of candidates attempted this question. Part a) was answered reasonably well, but the evidence suggests that a substantial number of candidates confused reusability with reuse.

Many candidates answering part b) produced adequate answers. This in particular applies to benefits of software reuse.

Part c) caused some problems – Section (i) in particular, as many answers were general and irrelevant. Section (ii) was answered reasonably well.

Section B

Question 4 [Software Process Improvement]

a) Define the term software process improvement, and explain how the process triangle of product, people and technology can impact quality and performance

(5 marks)

b) Give a brief explanation of what improvements can be made to the construction and infrastructure management processes of a traditional development process cycle.

(8 marks)

c) A small to medium sized software house is considering the use of a reference framework such as CMMI, and ISO/IEC 12207 for improving its own processes.

Write a report that presents a brief outline of ONE of these reference frameworks highlighting the degree of coverage of the software process, independence of specific methodologies, and acceptance amongst software professionals and communities.

(12 marks)

Answer Pointers

A good answer should cover the following points:

a) SPI is concerned with controllable factors within the software development process affecting software quality and organizational performance. As product become more complex the greater the challenge in reaching performance and quality targets; high levels of skilled and motivated people, should lead to higher quality and better organizational performance; the better the technology available in terms of engineering tools and methods the greater the productivity.

(5 Marks)

b) The “traditional” software development process is often represented by a sequence of phases from development into maintenance. Within each phase, there exist many activities that can be implemented to varying degrees. These improvements relate not just to the sequencing of the steps or to the inclusion (or exclusion) of certain steps but also to the specific implementation of the individual steps.

Improvements to the general area of requirements management, including capture, sign-off and change management by adopting prototyping methods or tools to a lesser or greater degree. In addition, improvements in the testing process can also yield positive benefits.

(8 Marks)

c) A good answer should present an outline of the selected framework, its scale, and applicability to the software house. For example:

- ISO/IEC 12207 - describes the continuing responsibilities that must be met and maintained during the life of the process. The processes involved in the life cycle of software development are categorized as either software-specific or system-context processes. The former is concerned with the activities directly related to the core software development effort, such as constructing detailed designs and writing codes; whilst the latter relates to the broader life cycle of systems development, such as project planning and systems operation.
- The framework identifies 43 software-specific and system context processes, with over 400 corresponding tasks.
- The highest level of organizational maturity is concerned with optimizing a process;
- As a large, formally developed framework, international recognition, standards, and acceptance are well established. However, whilst coverage of the software process is comprehensive, it may be somewhat challenging for the small to medium software house, and are likely to be tailored;

(12 Marks)

Examiner's Guidance Notes:

This question was the least popular of all questions and returned the second lowest pass rate of near 28%.

The evidence suggests that there was a lack of knowledge and awareness of Software Process Improvement and its associated reference frameworks. This extended across all parts of the questions including part a) definitions, b) application to traditional development lifecycle, and c) example frameworks. Answers were often insubstantial, were not relevant, failed to go further than part b), and of little academic merit in many instances.

Question 5 [Software Management]

The developer of an application which manages team-based projects, having previously operated as a sole trader, has seen a major growth in customer numbers. The decision has been taken to employ up to six members of staff for the continued development and maintenance of the application.

a) Write a report that:

(i) outlines to the former sole trader, the issues involved in selecting people for software development and maintenance work today;
(ii) discusses key issues and techniques for team organization, cohesion, and motivation.

(16 marks)

b) Briefly explain the People Capability Maturity Model (P-CMM) and discuss any similarity with aspects of the agile philosophy.

(9 marks)

Answer Pointers

A good answer should cover the following points:

a) Selecting and motivating people for software development and maintenance according to:

(i) Experience as a programmer, on various platforms, project team domain knowledge; educational background; communication and transferable skills. Today, there is a general shortage of experienced staff; can be expensive as remunerations need to be competitive; training should be viewed as long-term investment, but it's important that retention remains high before and after.

(ii) Maslow suggested a hierarchy of personal and individual motivational needs where social, self-esteem, and self-realisation needs typify high order needs. Most professional software is developed in teams. It is important that developers and bug-fixers needs are met within the work environment and therefore, their teams. However, for such teams to be effective, they need to have the right balance of skills, experience, and personality-type (such as Belbin test); share a common purpose; be in regular communication; and members roles are well-defined and valued.

(16 Marks)

b) The People Capability Maturity Model – a framework for improving the way in which an organization manages its human assets, a five-level model from initial to optimizing.

Its strategic objectives include increasing the capability of an organisation involved in software development by increasing the capability of its workforce; aligning the organization and its employee's motivation; and retention of critical human assets.

Similar concept in Agile philosophy in the areas of:

Individuals and interactions over Processes and tools

Working software over Comprehensive documentation

Customer collaboration over Contract negotiation

The agile philosophy re "Individuals and interactions over Processes and tools" gives emphasis to such things as: building projects around motivated individuals; their working environment and support needs; trust, autonomy, and self-organizing teams to get the job done.

(9 Marks)

Examiner's Guidance Notes:

This question was the second most unpopular of all questions and returned the lowest pass rate of near 24%.

Candidates demonstrated some understanding of Part a) even though some responses relied on common sense and experience rather than academic or empirical research.
The evidence suggests that many candidates showed little knowledge or awareness of the P-CMM and, those that went on to attempt Part b) focussed only on the agile philosophy and what had been committed to memory.