THE BCS PROFESSIONAL EXAMINATION
Professional Graduate Diploma

April 2015

EXAMINERS' REPORT

Software Engineering 2

General Comments

The pass rate was 46%. The following issues appear to be significant:

1. Coverage of the syllabus. Candidates are encouraged to adopt a staged approached to their learning, by completing the Software Engineering 1 module, before attempting Software Engineering 2. This is of particular concern, as much foundational knowledge would seem to be missing, resulting in candidates using common sense interpretation as a substitute for depth in basic principles and concepts of software engineering. Further, in terms of breadth of coverage, there is evidence that many candidates appear to have knowledge of traditional methods, but limited knowledge on modern methods, tools, and techniques identified in the new syllabus.

2. Subject awareness. A successful candidate 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 instances of candidates answering too many questions resulting in very "shallow" answers"; or too few, which resulted in too detailed an answer but insufficient to compensate for the marks lost by not completing another question, or 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.

Question 1

A software company engaged in Object Oriented development has been collecting both software product metrics and software process metrics over a number of years.

a) Explain what is meant by software product metrics and give three examples of such metrics relating to OO software.

(8 marks)

b) Explain what is meant by process metrics and give examples of three such metrics relevant to a software company's process improvement.

(8 marks)

c) Explain what is meant by the Goal Question Metric (GQM). Outline how GQM and both software product metrics and software process metrics can be used in a company's efforts to determine and to improve their software quality.

(9 marks)

Answer Pointers

A good answer should cover the following points:

a) Software product metrics measure properties and aspects of the software product itself, such as its size in lines of code, its complexity, no of functions, execution speed, etc. They can involve direct measurements or indirect measurements of product properties in the case of complexity. Three examples of product metrics which relate to OO software are:
(2 marks)
Number of Classes is a direct count of the number of classes in an OO system.
(2 marks)
Number of Children (NOC) is a direct measure of the number of direct descendants for each class.
(2 marks)
Depth of Inheritance Tree (DIT) is the depth of a class within the inheritance hierarchy defined as the maximum length from the class node to the root or parent of the hierarchy tree and is measured by the number of ancestor classes.
(2 marks)
b) Process metrics measure properties and aspects of the software processes during development and maintenance, either directly or indirectly, such as time taken by a team to perform an activity, team effort per activity, staff turnover on a project, etc. Examples of relevant process metrics with the aim of improving the process and quality of the software produced are as follows: (2 marks)

Measuring response time from customer report of software error to its resolution with an aim to improve response time of maintenance team.

(2 marks)

Measuring levels of communication between members of a distributed development team with the aim of improving team communication during development.

(2 marks)

Measuring use of CASE tools by the development during a process to determine which tools are effectively supporting the process.

(2 marks)

c) Goal Question Metric (GQM) proposed by Victor Basili is a method for defining and interpreting software measurements. It can be used to provide feedback and an evaluation of a software process or product. The starting point is to agree on the goal to be established for a software process or product. A goal could be to improve software reusability or improve on software delivery time. The next step is to determine questions by refining the goal into a set of quantifiable questions and then to define appropriate metrics which will allow these questions to be answered.

(6 marks)

GQM can be used as part of a software quality improvement programme as follows: firstly in a planning phase the current processes and products are evaluated and quantifiable goals for improvement are established; new methods or tools to achieve these goals may be introduced and their effectiveness is measured during development and post-development the results achieved are analysed and evaluated to determine the extent to which the goal has been achieved. Understanding gained can then be applied in future process improvements to improve their software quality.

(3 marks)

Examiner's Guidance Notes:

This question assesses a candidate's knowledge and awareness of software measurement concepts and processes.
Nearly 35 % of candidates attempted this question, but only some produced reasonable answers. There is evidence that most candidates provided proper explanations of software product and process metrics in parts (a) and (b), but many were unable to give proper examples of the corresponding metrics. A substantial number of candidates discussed software quality attributes (e.g. reliability, usability, etc.) and software process improvement instead.
Only a small number of candidates answered part (c) reasonably well.

Question 2

a) Explain what is meant by the phrase "software as a service" giving at least 3 examples of software services available today.

(8 marks)

b) As a Software Engineering consultant, you have been asked to advise a small company on the advantages and disadvantages of using an externally provided payroll system provided as a software service. In your answer, outline both the advantages and disadvantages to the company.

(12 marks)

c) As a software developer of a software service, discuss what steps you could take to overcome the disadvantages that you identified in b).

(5 marks)

Answer Pointers

A good answer should cover the following:

a) Software as a Service means access to software is provided to customers/users via a network, typically the internet, by a vendor or service provider who hosts the software remotely. The customers/users usually pay a subscription to have access to the software as a service on the remote host, so it can be thought of as a "pay-as-you-go" model of delivering software. It is sometimes also called "software on demand" as customers/users access the service on an as-needed basis; they do not need to buy, install and manage the software.

(5 marks)

An example of software as a service is a business application such as customer relationship management (CRM), offered centrally as a service, via web access, to a number of businesses that can use the CRM as needed. Many business applications are available as SaaS and this means companies can concentrate on their core business and using only the software services when they need them.

(3 marks)

b) There are many advantages that a small company can obtain from using an externally sourced payroll system provided as software as a service. They do not have to invest in developing their own payroll software or purchasing and maintaining the payroll software. They will not have the cost of extra hardware and staff to run the payroll software internally. They can concentrate on their core business functions and simply make use of the payroll software on an as-needed basis. The service provider is responsible for maintaining the software.

(6 marks)

Some disadvantages are loss of control and knowledge of the software. Their requests for changes to the software service may be ignored if they are too small users of the service and if they are not in line with the service provider's development plans. There is also a risk that data sent to be processed remotely may be insecurely handled by the service provider. The service provider may not provide details of how its software has been developed or how it is maintained so it may be difficult for customers to determine its quality before purchasing the service. The service provider's business could fail leaving the small company without any payroll service.

(6 marks)

c) The developers of a software service could overcome the disadvantages as follows:
In their contracts with users/customers, the provider could make clear the extent of customization that customers are allowed and state their policies on maintenance of the software underlying the service.
The developers could ensure that the underlying software has been developed to facilitate its future maintenance and enhancement.
The providers could also provide data concerning the quality, reliability and security of the service and guarantees of service.
The sources of the software underlying the service could be held in escrow and made available to customers in the event of the service provider going out of business.

(5 marks)

.

Examiner's Guidance Notes:

Nearly 90% of candidates attempted this question.
In part (a), there is evidence that many candidates confused "Software as a Service" (SaaS) with e.g. outsourcing or generally available 'services' such as on-line booking, on-line shopping, etc. Some candidates discussed also 'cloud computing'.
Part (b) was generally answered reasonably well. Many candidates however discussed advantages and disadvantages of externally acquired software instead.
Only a small number of candidates sufficiently answered part (c) by addressing disadvantages identified in part (b).

Question 3

a) Explain what is meant by an assertion in the context of software design and explain how the Object Constraint Language can be used to express the following assertions and, in each case, give an example:

i) Invariant property of a class
ii) Pre-condition of a method
iii) Post-condition of a method

(10 marks)

b) Outline three reasons why assertions are useful in software design.

(9 marks)

c) Discuss software verification and its role in ensuring the correctness of a software implementation.

(6 marks)

Answer pointers

A good answer should cover the following points:

a) An assertion is a boolean expression or predicate that describes a property of condition relating to the software design and that evaluates to *true* or *false*. The Object Constraint Language allows software designers to associate assertions with UML diagrams modelling classes and methods and with the corresponding source code, down to the individual program statement level.

(1 mark)

I. An invariant property of a class can be expressed as follows: Consider the following class Lecture with associated Duration: Int where all lectures must have duration of less than 45 minutes, we can express this as context Lecture inv: duration < 45

(3 marks)

II. A pre-condition of a method is an assertion that states what must be true before a method is executed. For example, before a Customer can withdraw money via a cheque from their account, there must be sufficient money in the account to cover the cheque.

```
Customer::withdraw(cheque)
Pre: accountBalance-cheque.amount > 0
```

(3 marks)

III. A post-condition of a method is an assertion that states what must be true after the method is executed. For example, after a Customer withdraws money via a cheque from their account, the amount remaining in their account will equal the balance before the withdrawal decremented by the amount withdrawn via the cheque.

```
Customer::withdraw(cheque)
```

Post: accountBalance = accountBalance@Pre - cheque.amount (3 marks)

b) Three reasons why assertions are useful in software design are as follows:

1) Assertions make explicit in formal language the logical properties of software and constraints on the software. Using Boolean Logic or Predicate Calculus, these assertions can be subjected to formal proof.
(3 marks)
2) Assertions assist developers to write correct software, since they specify the expected behaviour of the software to be implemented. The developer can include code within their implementation to check that the assertions are true.
(3 marks)
3) Assertions provide documentation of interfaces and can be used by developers when debugging. Assertions provide guidance to developers using an interface and to debuggers about what values to expect when debugging an interface error.
(3 marks)

c) Software verification is summed up as "Are we building the system right?". Verification of software entails checking that a software system conforms to its specification.
(3 marks)
Working from a correct specification, through correctness preserving transformations of a formal specification, it is possible to ensure the correctness of a software implementation. Thorough checking at each stage from specification through to design and coding also plays a role in ensuring correctness.
(3 marks)

Examiner's Guidance Notes:

Only a small number of candidates (appr.17%) attempted this question.
In part (a) most candidates sufficiently explained the meaning of invariants and pre and post conditions but did not adequately provide proper examples.
Part (b) caused problems. Many candidates did not answer this part properly and produced irrelevant answers.
Part (c) also caused problems. There is evidence that only a few candidates gave a proper definition/explanation of software verification. The role of verification in ensuring the correctness of a software implementation was sufficiently addressed by a few candidates only.

Question 4
B1.

a) Write a report that presents an overview of open source software engineering. The report should highlight some of the successful projects and discuss how one of these projects has been used alongside or instead of commercial and proprietary application software to great advantage.

(15 marks)

b) Discuss the extent to which open source has helped companies deliver software with higher developer productivity, improved product quality, and projects completed within set budget and timescales.

(10 marks)

Answer Pointers
The answer to section a) should:

- take the form of a report with appropriate structure including title, sub-headings including Introduction, Main body, and Conclusion or Summary;
- The Introduction should give a brief outline of open source and its emergence as a software engineering discipline. Attention should be given to clearly distinguish between OSS and OSSE.
- The Main body of the report could view software engineering from the viewpoint of methods, tools, and techniques. In this regard discussion may centre on such things as the open source approach to requirements engineering, project management, process design, development, testing and quality issues. Many success stories can be highlighted such as development tools (ECLIPSE), applications (Apache, MOZILLA). The many failures could also be highlighted.

(15 marks)

The answer to section b) should:

- take the form of a discussion wherein evidence is presented in respect of clear productivity gains, limited improvements in quality (depending on one's definition of quality), open-ended completion milestones.
- Productivity gains through the building of community of developers, and access to extensive and open sources of human expertise, and code libraries;
- Quality improvements using forums for continuous but informal structured walkthroughs, feedback, from many groups and individuals as "proof" readers or code inspectors;
- Deadlines and budgets are the most difficult to assess due to the nature of communities (primarily dependent on volunteers and interest groups), and the fact that in open source, development and maintenance are often fused together and difficult to separate.

(10 marks)

Examiner's Guidance Notes:

This question assesses a candidate's knowledge and awareness of OSSE. This was the third most popular question amongst candidates, having the third highest pass rate (29%). The candidate's responses can be categorised into three groups: a very small group that understood OSSE; a sizeable group with very good Open Source knowledge, but not the software engineering side; and a significant group that viewed the subject as pertaining to access to free software.
Question 5
B2.

a) Write a report that gives an overview of the Spiral software lifecycle model, and demonstrate how it might be used to plan, organise, and run new software development projects in commercial enterprises.

(15 marks)

b) Discuss the view that Agile methods are far more able to deliver high customer and worker satisfaction than their traditional counterparts.

(10 marks)

Answer Pointers

The answer to section a) should:

- take the form of a report with appropriate structure including title, sub-headings including Introduction, Main body, and Conclusion or Summary;
- The Introduction should give a brief outline of the Spiral Model and its origins.
- The Main body of the report presents the different aspects of the model. In particular, the quadrants of goal Determination, Evaluation and risk assessment, product Development, and Planning. The cycle moves through each quadrant in the form of a spiral until the product is finally delivered.
- Prototyping is an important tool in the early cycles as viability, risk, and ROI are assessed;
- Quadrants such as Development include activities such as: Create design, Review, Develop code, Inspect, Test product;
- Highlight the fact that new projects are high risk due to such things as speed of change in customer needs, and operating environment. Therefore, a lifecycle model is required which is able assess risk as an implicit part of model, ability to terminate projects when the business case no longer exist, and release working software in stages through incremental development and prototyping.

(15 marks)

The answer to this section should:

- Highlight the aspirations of agile methods and discuss whether evidence of practice agree;
- The manifesto: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan.
- Demonstrate that its use has expanded in deed, but only some aspects of its aspirations have been realized (such as quick delivery of software, and responding to change), but major problems remains in terms of application “junk” and the software “eco” system, its environment and “recycling”.

(10 marks)

Examiner’s Guidance Notes:

This question assesses a candidate’s knowledge of key developments in process models. This was the second most popular question, having the second highest pass mark (48%). The candidate’s responses can be categorised into three groups: a small group that understood both traditional and modern process models; a sizeable group with very good knowledge of either the Spiral Model, or agile methods; and a significant group that lacked the required knowledge, and tried to use other models (esp. “Waterfall”) in place of what was requested.