BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 5 Diploma in IT

Software Engineering 1

Examiners' Report March 2017

General comments on candidates' performance

This examination had a slightly lower average mark than is normal for this paper.

Although there were some excellent papers, a large number of candidates lost marks needlessly by providing answers that contained minimal content or were unrelated to the question. Candidates are therefore advised to spend more time reading and understanding the requirement of each question before writing their answers.

Furthermore, candidates should be reminded that a basic level of English comprehension and an appropriate level of Technical English is essential for success in this paper.

Section A

A1. The assessment of software quality is a subjective process where the quality management team has to use their judgment to decide if an acceptable level of quality has been achieved.

a) Give THREE examples of questions related to a system's quality characteristics you would ask as a member of the quality management team to determine whether or not the software is fit for its intended purpose.

(6 marks)

Answer Pointers

Examples of questions related to a system's quality characteristics that could be asked by a member of the quality management team when considering whether or not the software is fit for its intended purpose:

- Have programming and documentation standards been followed in the development process?
- Has the software been properly tested?
- Is the software sufficiently dependable to be put into use?
- Is the performance of the acceptable for normal use?
- Is the software usable?
- Is the software well structured and understandable?

2 marks for each question given

b) Software quality is not just about whether the software functionality has been correctly implemented, but also depends on the software quality attributes for example dependability, usability, efficiency and maintainability. List another SIX quality attributes.

(6 marks)

Answer Pointers

Software quality attributes include:

- Safety
- Security
- Reliability
- Resilience
- Robustness
- Understandability
- Testability
- Adaptability
- Modularity
- Complexity
- Portability
- Usability – already covered in the question
- Reusability
- Efficiency – already covered in the question
- Learnability

c) It is not possible for any system to be optimised for all quality attributes.

Select TWO quality attributes and explain what factors may have to be sacrificed when these quality attributes are improved.

(4 marks)

Answer Pointers

Factors that may have to be sacrificed when quality attributes are to be improved would be similar to:

- improving robustness may lead to loss of performance,
- improving efficiency may lead to loss of adaptability.

d) Commercial pressure for an early product release will affect product quality.

i) State THREE quality attributes you might pay less attention to in these circumstances.

(3 marks)

ii) When less attention is paid to EACH of your quality attributes stated above, identify TWO characteristics of the final product that may be affected as a result.

(6 marks)

Answer Pointers

Quality attributes that might receive less attention when commercial pressures for an early product release are applied would be similar to:

- paying less attention to modularity at the risk of affecting testing and reusability in the final product;
- paying less attention to usability at the risk of affecting customer acceptance and the need for additional training;
- paying less attention to efficiency at the risk of affecting response rates and maintainability.

Examiner's Guidance Notes

This was the second most popular question in Section A.

Many candidates gave questions relating to requirements elicitation rather than software quality.

Most candidates were able to identify four software quality attributes, although some reiterated those qualities embedded in the question.

Most candidates were able to explain what factors may have to be sacrificed when particular quality attributes are improved.

Most candidates were able to identify three quality attributes that may pay less attention to when under commercial pressure for an early product release.

Only some of these candidates put forward how they considered other quality attributes may be placed at some risk in this situation.

A2.

a) Give THREE reasons why, in the start up phase of a project, estimates will have a wide margin of error.

(6 marks)

Answer Pointers

Reasons why, in the start up phase of a project, estimates will have a wide margin of error:

- High-level user requirements definition
- Software having to run on unfamiliar computers
- Software having to use new development technology
- People involved in the project and their skills will probably not be known
- Difficulty in assessing the accuracy of different approaches to cost and effort estimation

2 marks per reason given

b) Experience-based estimating techniques can be used to make software effort and cost estimates. Describe THREE activities a manager might perform when applying these techniques.

(6 marks)

Answer Pointers

Activities a manager might perform when applying experience-based estimating techniques:

- Identify the deliverables to be produced in a project
- Identify the different software components or systems that are to be developed
- Document the above in a spreadsheet, estimate them individually, and compute the total effort required
- Get a group of people involved in the effort estimation and to ask each member of the group to explain their estimate. This often reveals factors that others have not considered, and you then iterate towards an agreed group estimate.

2 marks per activity listed

c) COCOMO-II is a model used to help estimate the effort and cost of a software product by using cost drivers to determine the amount of effort required. Describe how each of the following cost drivers can influence the amount of effort that will be required in the development of a software system:

(i) Reliability **(3 marks)**
(ii) Reusability **(3 marks)**
(iii) Analyst capability **(3 marks)**
(iv) Multisite development **(2 marks)**
(v) Development schedule **(2 marks)**

Answer Pointers

How each of the following cost drivers can influence the amount of effort that will be required in the development of a software system:

i) Reliability - This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience then the effort required is low. If a failure would risk human life then the effort required is very high.

ii) Reusability - This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.

iii) Analyst Capability - Analysts are personnel that work on requirements, high level design and detailed design. The major attributes that should be considered in this rating are Analysis and Design ability, efficiency and thoroughness, and the ability to communicate and cooperate.

iv) Multisite Development - Determining its cost driver rating involves the assessment and averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia).

v) Required Development Schedule – if there is a need for rapid delivery more effort will be required. Accelerated schedules tend to produce more effort in the later phases of development because more issues are left to be determined due to lack of time to resolve them earlier.

Examiner's Guidance Notes

This was the least popular question in Section A.

Most candidates provided three reasons why in the start of a project estimates will have a wide margin of error.

Few candidates were able to describe three activities a manager might perform when using experience-based estimating techniques.

The majority of candidates answered the five COCOMO-II questions by describing what they understood by the meaning of reliability, reusability, analyst capacity, multisite development and development schedule rather than what the question was asking which was how each of the cost drivers can influence the amount of effort required.

A3.

a) Explain how prototyping can be used in the following software development processes:

(i) Requirements engineering process **(3 marks)**
(ii) Systems development process **(3 marks)**

Answer Pointers

i) Requirements engineering process:

- A prototype can help with the elicitation and validation (meets the real needs of the client) of system requirements.
- It allows users to see how well the system supports their work.
- They may get new ideas for requirements, and find areas of strength and weakness in the software.
- They may then propose new system requirements.
- As the prototype is developed, it may reveal errors or omissions in the requirements that have been proposed. A function described in a specification may seen useful and well defined. However, when that function is combined with other functions, users often find that their initial view was incorrect or incomplete. The system specification may then be modified to reflect their changed understanding of the requirements.

ii) Systems development process:

- A prototype can be used to explore particular software solutions
- A prototype may be used while the system is being designed to carry out design experiments to check the feasibility of a proposed design.
- Because of the dynamic nature of user interfaces, textual descriptions and diagrams are not good enough for expressing the user interface requirements.
- Rapid prototyping with end-user involvement is the only sensible way to develop graphical user interfaces for software systems.

b) Describe what is meant by the term *throwaway prototyping*.

(3 marks)

Answer Pointers

Throwaway prototyping (3 points from, or similar to):

- With 'throw-away' prototyping a small part of the system is developed and then given to the end user to try out and evaluate. The user provides feedback, which can quickly be incorporated into the development of the main system.
- The prototype is then discarded or thrown away. This is very different to the evolutionary approach.
- The objective of throwaway prototyping is to ensure that the system requirements are validated and that they are clearly understood.
- The throwaway prototype is NOT considered part of the final system. It is simply there to aid understanding and reduce the risk of poorly defined requirements. The full system is being developed alongside the prototypes and incorporates the changes needed.

c) Describe what is meant by the term *evolutionary prototyping*.

(3 marks)

Answer Pointers

Evolutionary prototyping (3 points from, or similar to):

- Software that is developed not only to help us answer questions but also to be incorporated into the final product.
- The idea behind this is that an initial prototype is presented to the user. They provide feedback and suggestions for improvements.
- These are actioned by the developer who then presents a more refined prototype. The user once more provides feedback. The process is repeated.
- So at each stage the prototype 'evolves' towards the final system.
- We have to be far more careful in its development, because this software has eventually exhibited the quality requirements of the final product, and these qualities cannot be retrofitted.

d) Discuss TWO major uses of using paper-based prototypes in the development of a software product.

(4 marks)

Answer Pointers

- To communicate ideas: between designers, developers, users and other stakeholders in the first stages of the user-centered design process.
- As a usability testing technique: to observe the human interaction with user interfaces even before these interfaces are designed and developed.

2 marks for each use cited

e) Compare the advantages and disadvantages of using prototyping in software development.

(9 marks)

Answer Pointers

9 points from the lists below, to include at least one from each list):

Advantages:

- Better quality system delivered: Sometimes a developer may not fully understand what the end user is expecting. Prototyping enables any misunderstandings to be identified and sorted out early on in the process.
- Identify problems early on: A working system is available early on in the process. The user can identify possible improvements that can be made before the system is completed.
- End user involvement: The end user feels more involved in the development of the system and will 'buy' into it.
- Fulfil user requirements: A system that has been through prototyping will generally have an improved design quality and will be far closer to what the user needs.
- Cost savings: It is far less expensive to rectify problems with the system in the early stages of development than towards the end of the project.
- Training: The prototype can eventually be used to help train staff while

the real system is still being fully developed.

Disadvantages

- Excessive development time: When the end user is asked to evaluate a prototype and provide feedback there is a risk that they will be forever wanting to tweak the system, thus leading to delays in development.
- User confusion: Sometimes features appear in a prototype, which are then removed, from the final system. Users can become confused or disappointed with the final system if it differs greatly from the prototype.
- Increased development time: The end user might start to ask for features to be included which were never in the original user requirements specification. This can lead to increased development time and costs.
- Too much focus on one part of the system: When a lot of time is spent on a prototype of one specific part of the system, other parts of the system might end up being neglected.
- Expense of prototyping: Building a prototype costs money in terms of development time and possibly hardware. While the prototype is being worked on, the real system is on hold.

Examiner's Guidance Notes

This was the most popular question in Section A.

Most candidates were able to explain how prototyping can be used in the requirements engineering process.

Fewer candidates were able to explain how prototyping can be used in the systems development process.

Candidates described several aspects of throwaway prototyping, but were a little less confident with evolutionary prototyping.

Only a few candidates were able to describe two uses of paper-based prototyping; the majority of candidates instead described the use of paper-based documents within software development life cycles.

Most candidates were able to articulate several advantages and disadvantages of using prototyping in software development.

Section B

B4. Incremental methods can be used in the development and delivery stages of a software project.

a) Describe any TWO advantages in using an incremental development method.
(6 marks)

Answer Pointers

Any two advantages that stress that deliverable portions are arrived at more quickly rather than waiting for full implementation and can enhance requirements refinement. Typically from; Cost of varying requirements is reduced as they are more easily accommodated. Simplifies obtaining customer feedback on up to date production, easier for clients to see progress of development. Faster delivery to customer of useable software products. Useable functions can be amended and updated quickly. Simply naming will attract 1 mark each with further 2 marks for a description

b) Describe any TWO disadvantages in using incremental development.
(6 marks)

Answer Pointers

Any two disadvantages. Typically, Low visibility caused by overheads incurred in producing documentation, producing plans, producing records in increments that are then changed. Integration impeded by difficulty in applying changing increments to overall architecture. Significant issues are: Software breakage, later increments requiring change to previous increments, no focus, low portability, and lower scalability. Less efficient caused by fixed overhead costs only apportioned over a small subset of the project. Simple naming 1 mark up to a further 2 marks for a description.

c) Explain any TWO advantages of using incremental delivery in a project.
(8 marks)

Answer Pointers

Any two advantages typically, clients can use the early increments as prototypes better understanding an evolving requirement set, increments can provide early training in aspects of product introduction gaining familiarity with functionality and interfaces. Clients gain more immediate value from an increment rather than wait for completed product. Increments can be designed to address most critical aspects of requirements early. Clients see a more immediate return on investment, possible to prioritise to meet client expectations up to 4 marks for each explanation. Simple naming 1 mark.

d) What type of project is not suited to incremental methods?
(5 marks)

Answer Pointers

Those projects that are very large, complex infrastructure, large distributed programming teams, those that require more sophisticated team management.

Where incremental delivery might cause disruption and decrease integrity of whole system. Projects that have a need for reconfiguration and documentation or similar.

Examiner's Guidance Notes.

This was the most popular question in part B with 89% of candidates making an attempt. Very few candidates achieved high marks, with approximately 18% achieving more than half of the available marks for the whole question.

There was often no distinction made between incremental delivery for development and incremental delivery of a project and answers often repeated the same material for part a) and c).

Most candidates did well in part d). In general, there was a lack of knowledge of the use of incremental methods – with a few exceptions where some candidates displayed a thorough understanding of benefits/disadvantages of the approach.

B5.

a) A software product is more effectively tested if it is built with testability as a consideration. Describe any FOUR characteristics of a software product that will enhance its testability.

(12 marks)

Answer Pointers

Four from Bach's list or similar (see Pressman CH 18) with typical (non -exhaustive) descriptions based on the following:

Operability: "The better it works the more efficiently it can be tested"

- System has few bugs (bugs add overheads)
- Bugs don't block test execution
- Product built in functional stages.

Observability : "What you see is what you test"

- Distinct output for distinct input
- Past System states and variables are visible during execution
- Incorrect output easily identified
- Internal errors automatically reported
- Source code is acceptable.

Controllability: "The better control the more testing can be automated"

- All possible outputs can be generated by some input combination
- All code is executable through some input combination
- Test can be conveniently specified, automated and reproduced

Decomposability:" By controlling the scope of testing we can quickly isolate problems and do more efficient testing"

- Implies that the system is composed of independent modules
- Modules are capable of independent testing

Simplicity: "The less to test the more quickly it is tested"

- Features are minimum to satisfy requirements

- Architecture is modularised to provide structural simplicity
- Adherer to a code standard this eases maintenance and inspection

Stability: ”Fewer changes allow fewer disruptions to testing”

- Infrequent software changing
- There is a control applied to change
- Changes should not invalidate existing tests
- There should be strong recovery mechanisms from failure

Understandability: “The more information we get the better we can test”

- The design should be well understood
- All dependencies between internal and external interfaces are well understood
- All design changes are well understood
- Technical documentation is accessible
- All technical documentation is organised, specific, detailed and accurate.

b) Briefly explain ways in which visibility can be enhanced in software testing.
(7 marks)

Answer Pointers

Visibility can be achieved by ensuring the developers understand the reasons and purpose of the tests, making them more likely to accept the results. Developers should be made aware of the test methods and understand the underlying technology function and operation possibly through training if required or through adequate documentation of the tests. Tests should be characterised to show developers explicitly the outputs e.g., black box for operating environment, white box for internal structures, integration test for modules acceptance testing for packages. The purpose and outputs should well-understood by the development team. Tests should be repeatable, non-repeatability of tests contribute to lack of visibility. Visibility is made poorer by poor documentation, poor plans and procedures and badly designed content management for project records as they obscure underlying process. 3 marks for a description only. Up to further 4 marks for demonstrating understanding of other factors.

c) The validation and verification process ensures that software is ‘fit for purpose’. Describe the difference between validation and verification.
(6 marks)

Answer Pointers

The difference (from Boehm):

Validation: are we building the right product?

Verification: are we building the product right?

Verification has focus on checking that the product specification is being met through the functional and non-functional requirements. Software verification involves such things as testing and bug fixing, unit tests, integration tests etc. Satisfying requirements and design criteria.

Validation is checking that the product is actually what the client wants. Validation testing is ensuring the product conforms or meets customer expectations acceptance tests are part of validation testing etc.

1 mark each for the terse definition only up to a further 4 marks for explaining the difference. Simply extending a definition of both is a maximum of 4 marks

Examiner's Guidance Notes.

This was the least popular choice for Section B. With only 62% of candidates making an attempt. The number of candidates achieving better than half of available marks for the question was low with less than 13% gaining more than half. The overall pass rate was slightly better than B4.

Many candidates were not able to articulate a description of testability of a software product but did give a description of test techniques and methods and subsequently gaining few if any marks in part a.

In part b), concerning visibility, a significant number of candidates explained how the brightness/colour schemes of the interface would help in making it more visible. The evidence suggested that many candidates had no exposure to the basics of the concepts surrounding the importance of visibility of testing to the development team (whose work is being tested).

Part c) was answered by almost all candidates with a few attempts that confused the differences.

B6.

a) When planning to adopt a Reuse-based software development project there are several key factors which should be considered. Explain any THREE of these factors.

(12 marks)

Answer Pointers

Need for quick development /tight schedule indicates a positive for reuse of software. The equivalent to requirements might be slightly mismatched but the benefit of the speed of matching outweighs disadvantages. Minimising development time is a key factor.

Expected lifetime of the product. Long life systems mean a key factor is maintainability. This issue includes considerations of changes change control, need for source code access, API access. Lack of future vendor support is a key factor in long life expected applications.

Sophistication of team. Reuse will need expertise in the technical domain to at least the level of the product being used. Low level code requires skilled developers so focus on reusing software that more closely matches.

The need for external audit certification. If your product needs verification the reused code might not be available/up to audit standards/certification. This would also apply to critical code.

If reusable code is available for your specific domain then is a positive driver to reuse.

Platform dependency-some code will only run on a specific platform such as Microsoft or Unix or have version dependencies that do not match your target users. These indicate possible disadvantages

Any three explanations from those above or similar lists (see also, Somerville/Pressman) 2 marks for a simple list, 4 marks for an explanation of any three

b) Using a commercial-off-the-shelf (COTS) package is an example of software application system reuse.

(i) List THREE benefits of using a COTS package. **(3 marks)**
(ii) List THREE disadvantages of using a COTS package. **(3 marks)**

Answer Pointers

i) Benefits
Rapid deployment
Easy to judge functionality
Avoid development risk
Reduce resources for development
Maintenance responsibility of vendor
1 mark each for any 3 from above or similar

ii) Disadvantages
Need to adapt requirements
Client needs to adapt to the package
May not be well enough documented
Dependent on cots vendor for development advice
Vendor might go out of business
1 mark each for any 3 from above or similar

c) Describe TWO important problems encountered when attempting to integrate a COTS package with other packages and systems **(7 marks)**

Answer Pointers

Lack of control over both functionality and/performance might be poorly implemented. The priority of vendor and integrator may not match. When COTS stands alone it has known function/operation when integrated with disparate system might be aberrant in behaviour little incentive for cots vendor to fix such problems or to identify which vendor or other package is not behaving correctly

Interoperability

Difficult to get different COTS to work seamlessly Vendors will have their own standards/protocols. Vendors have expertise only in their own domain might not be able to discuss the integrated operating environment. Lack of common standards and cooperative version control/updating/upgrading

Lack of control

No control over update cycles maintenance upgrades or protocol changes /updates to base package difficulty in maintaining common compatibility between different vendors upgrade versioning schedules. System evolution difficult to maintain control. Over time support may be withdrawn making operation of other packages impossible/difficult to budget for such planning contingencies

Vendor support

Vendor may go out of business no accesses to source code documentation/ API's. One cot might dominate usability of the whole integrated system with no possibility of further support Often vendors are taken over by other business changing product offering.

2 marks for each basic description a further 3 marks for making important points in description

Examiner's Guidance Notes.

This question was the second most popular in Section B. Well over a third of candidates achieved better than half of available marks for the question. In general, most of the candidates felt able to attempt all parts with varying degrees of success. There was a significant range of marks between candidates with many achieving very high marks and many only gaining very low marks.

Part a) was often misinterpreted as a question purely about COTS. Those candidates that understood the development context of code reuse did well throughout the whole question. Those candidates that had little concept of the development context tended to view the question as a management issue surrounding COTS and subsequently attempted part a) and part c) with the same answer being simply replicated and often vague.

Part b) was in general well answered (it required a simple listing), although some candidates confused benefits and disadvantages.