# BCS PROFESSIONAL EXAMINATIONS
# BCS Level 5 Professional Graduate Diploma in IT

**April 2008**

# EXAMINERS' REPORT

# Software Engineering 2

## General Comments

A better set of results produced this year, which may be due in part to better preparation of students for examination, and exam questions that were more structured and less open to individual interpretation. Problems still persist with a significant minority of students answering more questions than required; poor layout of answers and illegible handwriting; the level of comprehension of the subject matter as expressed in their use of English.

Questions of a technical nature (those questions requiring the students to demonstrate ability to apply and use techniques), continue to be unpopular and candidates have done quite poorly again. Thus, the structure of the majority of questions was such that the first part of a question tested a candidate's knowledge of the subject; and the second part sought to test candidates' ability to show selection, prioritisation and critical judgment and application of this knowledge in the scenario presented. Many answered Part A reasonably well based on the recall of facts. However, Part B was either not attempted or the answers given made no reference to the scenario. Thus, without evidence of critical judgement and relevant application, these answers were given low marks. To address the lack of depth in answers, it is important that candidates recognise the importance of study through reading more widely in terms of publications within the profession, as well as the recommended text.

Question 1 and 4 proved to be two most popular choices amongst candidates, and exhibiting the highest pass rates. Question 3 was the least popular question and had the lowest pass rate of all. It is clear from the answers supplied in question 3, that many candidates lacked the breadth of knowledge and general awareness of the principles that underpin the engineering of software, and the purpose and aspiration of formal methodologies. Thus, they were unable to analyse and synthesise requirements, and identify the appropriateness of tools for scenario described.

**Question 1**

a) Differentiate clearly, with reasons and examples, between
- Verification and validation
- Quality assurance and quality control
- Configuration audit and quality audit.

**(12 marks)**

b) "Software quality processes are often ascribed only to development activities, yet the biggest impact of quality software is during the maintenance phase." Discuss the statement, making clear whether you agree or disagree, with full reasoning.

**(13 marks)**

**Answer Pointers**

A good answer should be structured in the following manner:
a) Verification is testing the software build process and its results to ensure they accurately mirror the design and standards used in the process model. Validation involves testing with the client or testing by the client that the product meets client's needs. 2 x 2 marks

Quality Assurance is a set of promises to conduct the development process in a particular visible and transparent way. Quality Control is the execution of those promises and the recording of executions with quality records. 2 x 2 marks

Configuration Audit is a functional test of a software configuration to ensure compliance with requirements. Quality Audit is a verification of compliance with Quality Assurance (usually by inspection of quality records accumulated by Quality control). 2 x 2 marks

b) The candidate should lay out a case that examines how software development processes are conducted during development (4 marks) ,
then what issues arise after deployment (whether adaptive, perfective or corrective)
(4 marks)
and then make the link between the maintenance activities and the quality records (such as design specifications, interface specifications, test specifications) accumulated during development. (5 marks)

This style of reasoning, or similar, with traceability of conclusions from arguments, for maximum marks.

**Examiner's Guidance Notes**

This question assesses the candidate's knowledge and awareness of software quality issues and their inter-relationships.

In Part (a), many candidates chose to describe Configuration Management as a complete entity, and offered much that earned no marks. Some gave tautological answers ("Verification means verifying the software.") Sadly, many candidates did not appreciate the inter-relations between these issues of Quality. Quality assurance and quality control are software engineering disciplines, not delegated to users, as implied by some answers.

In Part(b), the question sought evidence-based reasoning to link reduced cost of maintenance with a rigorous quality regime during development. Many candidates did not link the two phases of development and maintenance. Very few candidates realised that documentation produced during development (for design, or test specificiations, or test results) could be used as a time-and-cost-saving guide during maintenance. A better appreciation is expected of the inter-relationships among quality disciplines and the way they combine to deliver software with quality.

**Question 2**

a) Compare and contrast the methods of software project estimating known as *size-related estimates* and *function-related estimates*.

**(7 marks)**

b) A software company has asked you to create a process improvement programme for them.

Derive THREE categories of activity classifications you would use. Briefly explain the reasons for your choices.

**(6 marks)**

Derive THREE types of process metrics you would use. Briefly explain the reasons for your choices.

**(6 marks)**

Discuss TWO critical success factors that will determine a good outcome from this programme, with your reasons.

**(6 marks)**

**Answer Pointers**

a) A good answer should describe Size-related measures as most commonly measures of Lines of Delivered Source Code (LOC). Other acceptable measures are pages of documentation, thickness of documentation, or number of object-code instructions. (2 marks)

Function-related measures relate to overall functionality of delivered software. Function points are the best known of this type. (2 marks)

The answer should go on to 'compare and contrast' these two systems. For example, estimating a size measure at the start of a project is usually by comparison with other, similar completed projects. The difficulty is finding something already completed to act as a guide. Estimating this measure at the start of a project may be by counting features in the requirements definition, or waiting until a high-level design exists then counting the number of design elements that need to be produced as code.

The difficulty is knowing the rate of development by your own people with new tools, or the rate of development by new people with old tools, or similar combinations of uncertainty. (3 marks)

b) A good answer should derive three types of **activity classification**. For example, one way of tackling this is to choose any three of classic quality-cycle processes of plan-do-check-act, or to select three phases of the development life-cycle. Reasons for this choice should be made clear. For example, 3 x 2 marks

Planning - estimating activity durations.

Doing – software lifecycle development activities (requirements capture, design, code)
Checking – test case design, test case generation, testing.
Acting – process- and people-intensive activities such as reviews and inspections.

**Process metrics** should reflect the type of activity classes chosen. Examples are requirements cross-referencing, design corroborations, code reviews, test error counts, installation error counts, or maintenance error counts. 3 x 2 marks.

**Critical success factors** are many – appropriate judgements will be made. Factors such as management buy-in, people support (Support Groups and evangelists), attitudes (getting away from blame or punishment management cultures) are examples of factors expected. 2 x 3 marks.

**Examiner's Guidance Notes**

This question assesses the candidate's knowledge and awareness of project management and process improvement.

In Part (a), many candidates delivered excellent answers. A few delivered poor answers. There was no half-way house.

In Part (b), the question asked how to deliver a process improvement programme. Many candidates offered a full description of software metrics, whether for process improvement or not. It was also popular to give the results of improvement as critical factors necessary for improvement to happen. Some answers averred that quality assurance delivered improvement. It is the analysis of quality records to identify processes that fail more often than not that gives the goals for an improvement programme. Quality and improvement are two separate concepts.

## Question 3

Software engineering, like any other mature engineering discipline needs key principles that underpin methods, techniques, methodologies and tools used in processes to create products.

a) Define the three principles of modularity, abstraction, and generality. Discuss how each of these principles impact and benefit software processes and products;

**(15 marks)**

b) Briefly discuss the success of software engineering as an engineering discipline in terms of the principle of "rigour and formality".

**(10 marks)**

**Answer Pointers**

A good answer should be structured in the following manner:
a)
(i)
Definitions of modularity, abstraction, and generality that encompasses such things as: the decomposition of a problem into simple, self-contained components; being able to identify important facets of a problem whilst ignoring the detail; and being concerned with creating solutions that are reusable;
(ii)
The impact (positive or negative) and benefit considered within the context of the quality (intrinsic or extrinsic) of the development process and the final product. For example, modularity has a positive impact on process by yielding benefits such as increased management visibility; and product benefits such as maintainability.

b)
The discussion should relate to formal methods within the context of mathematically based tools and techniques for the software engineer to use in the specification, development and verification of software systems.

Success in “rigour and formality” might be considered significant in the context of semi-formal tools such as DSDM and UML, etc. However, unlike traditional engineering, these tools have limited universality as there does not appear to be any consensus on the meaning of formality and metrics. Further, there is limited adoption worldwide and often within specialist user groups only.

**Examiner’s Guidance Notes**

This question assesses the candidate’s knowledge and awareness of principles for software engineering, formal methods, and its universality within the profession.

This was the most unpopular question. Disappointingly, it had the worst pass rate. Most students understood the meaning of modularity, but were not clear on such things as abstraction and generality. What is clear from the answers submitted was a general lack of knowledge of the concept of formal methods, especially those opting to complete Part A only!

It is hoped that in future candidates will have a better appreciation of engineering principles for software, and knowledge of formal methods.

**Question 4**

Software development is characterised by a number of phases, the order and duration of which are determined by software process models.

a) For the incremental and spiral models, discuss how the development phases are ordered, and explain the means by which decisions are made to progress from one phase to the next.

**(15 marks)**

b) A major retailer requires an online shopping web application to gain a commercial advantage over its competitors. However, the application should run alongside existing legacy systems.

Evaluate the ability of the incremental and spiral process models to deliver the software that satisfies the user's functional requirements, and within timescale.

**(10 marks)**

**Answer Pointers**

A good answer should be structured in the following manner:

a) INTRODUCTION. Overview of software development lifecycle: Definition phase (analysis, design), Development phase (design, implementation, testing), support phase (release, maintenance);

DISCUSSION OF METHODS:

i) Incremental model: project broken down into functions with lifecycles of shorter duration; on completion of a function or a particular phase, other functions can be assessed and begin assuming resources available and the business case can still be justified; in contrast,
ii) Spiral. Makes use of a combination of models (such as incremental). Project spiral outwards through four-phase quadrant of: Objective setting, Risk Analysis, Development, Planning. The assessment of risk determines whether the project spirals any further towards the end goal.

b) EVALUATION OF METHODS:

In this application, a product needs to be completed quickly. Both methods would prove effective in dealing with changing user requirement and minimising risk.
The incremental model might prioritise the functionality of the online shopping application; and two to four weeks may be given to the development and release of each function.

However, whilst the Spiral model might adopt an incremental approach within its overall framework, risks are explicitly assessed during each loop, re the whole project.

A typical overall assessment might state that very large, high risk applications (such as a new online shopping facility in a very competitive market) might be served better by using the Spiral model;

**Examiner's Guidance Notes**

This question assesses knowledge and awareness of process models, and ability to select and justify choices within a given application context.

This was the second most popular question, with the second highest pass rate. Many candidates did not set a general SDLC context within which to present their answers but rather moved directly to describe the specific models. In part A, candidates were generally very knowledgeable about the Spiral model, but appeared have difficulty addressing the specifics of the incremental process model but gave answers pertaining to approaches as DSDM and Prototyping.

The answers submitted for Part B, demonstrated a general lack of ability to identify aspects of the scenario that would favour a particular process model over another.

It is hoped that future delivery of the subject will enable candidates to gain a better appreciation of a range of process models and enable them to identify the most suited application context for any given model.

**Question 5**

A busy independent nursing practice has asked you to develop an appointment booking system for use by both the receptionist and nurses belonging to the practice.

The system should be able to accommodate new patients, and give priority to patients according to their actual need.

Use an appropriate modelling technique with supporting notation (state any assumptions made) to produce the following:

a) Specification of the problem requirements in the form of a table that clearly identify the entities and the processes;

**(5 marks)**

b) Analyse the problem by identifying the input and output data and processes, tabulated in the form of typical data dictionary entries. These may include entities, their attributes, and processes and their parameters.

**(6 marks)**

c) A diagrammatic representation of the behavioural aspect of the system.

**(7 marks)**

b) A diagrammatic representation of the structure and relationships within the system

**(7 marks)**

**Answer Pointers**

A good answer should be structured in the following manner:

a) A table with say four rows by two columns, appropriately labelled, that identifies four key entities (Patient, Nurse, Receptionist, Practice), with their respective processes. For example, the Practice may have processes such as: *getOpeningHours(), getCurrentLocation(), getOutOfHoursContactNumber().*

b) A table with 6 to 10 rows by four columns including the following descriptive labels: *Name, Data Type, Security Level, Description*;
For example, a data dictionary entry for "patient name" using the above headings would take the form:
*Patient_Name, Attribute[String], private, "This represents the patient's initials and surname"*

c) The behavioural aspects of the system can be represented in a variety of ways. These include data flow, state transition, interaction, diagrams. The important behaviour to capture include:

*The Patient, makes appointment;*
*Receptionist adds new record ;*
*Nurse/Practice assigns priorities to patient.*

d) The structure and relationships within the system can be represented using ERD, Use Case, or Class diagrams. ERD relationships could include:

*Receptionist to Patient (one to many)*
*Nurse to Patient (one-to-one)*
*Practice to Nurse (one-to-many)*

**Examiner's Guidance Notes**

This question assesses a candidate's knowledge and ability to apply analysis/design methods and techniques to produce appropriate representations of the scenario, the problem depicted, and its possible solution.
This question was the second most unpopular with the third lowest pass rate. It was clear from the answers submitted that many candidates had a reasonable, knowledge of some parts of the method (or technique) but lacked an appreciation of the different views and the respective justification for having different model views. Many answers didn't seem to appreciate the format of a table, and typical data dictionary information entries for such things as software objects.