THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2005

EXAMINERS' REPORT

Software Engineering 1

General

This was the second sitting of this examination. Candidates mostly avoided the temptation to write a lot of descriptive material and focussed on the evidence-based reasoning that the examiners were seeking. This Diploma-level exam asks for some reflection but more application, synthesis and analysis.

Question 1

1. Software Engineers often use models (such as DFDs, or ERDs) as predictors of the final software product in the same way as other engineers use theory to predict the properties of their products. Discuss this concept and consider if models describe properties of software, or describe steps for an engineer to follow when developing the software. Give full reasons for your conclusion.

(25 marks)

Answer Pointers

A good answer should be a balanced answer, starting with some discussion of using DFDs and ERDs and describing them as possible design notations for to aid process or possible tools to predict product quality. (10 marks)

Then a reflection on design models, or transformations through the life cycle, versus theory that predicts properties. A very good answer might mention that software properties are defined in ISO9126 – reliability, maintainability etc., and that no design notations could predict any of those properties.

Based on this argument, the software engineer uses notations to guide the development process, not to predict the product's properties. **(15 marks)**

An alternative approach might start with considering how DFDs and ERDs are used (10 marks), and find we as yet only have models, rules and heuristics. **(15 marks)**

Examiner's Guidance Notes

This is what is called an unco-operative question. No breakdown of marks is supplied, so the candidate should reformulate the question so that a structure can be made that is likely to earn the points. A likely reformulation should look at the structure of the questions, and create a structured answer.

The question offers models (DFD, ERD). It mentions that other engineers use theory to create the product. It then suggests that software engineers might not have theory but might use models to 'move the development along'**.** An answer shape appears: first, describe what DFDs and ERDs are used for. Then, observe they are used in a particular transformational stage, and the software lifecycle consists of transformations through requirements, design code etc. Then, observe that they assist transformation, rather than model the properties of the software. The conclusion now appears; these models assist the process of development, they are not predictors of software properties.

This answer takes 4 steps, roughly 6 marks per step.

Few answers followed the model above. Many answers did not make any distinction between process and product.

Question 2

2. *a)* The name 'CASE' is sometimes used to mean Computer Assisted Software Engineering and other times to mean Computer Assisted Systems Engineering. Briefly analyse these terms and explain if you think there is a difference between them. **(5 marks)**

b) Discuss the 'software' and 'system' in the following scenario. In your answer, select two 'software' issues and two 'system' issues'. Give your reasons for selecting and categorising each issue. **(20 marks)**

The implementation of a distributed object architecture requires middleware (object request brokers) to handle communications between the distributed objects. In principle, the objects in the system may be implemented using different programming languages, may run on different platforms and their names need not be known to all other objects in the system.

Answer Pointers

a) A good answer will remark that 'System' implies something bigger than the software in which the software will operate [Somerville p12 Pressman p24]

Systems engineering entails a specification of a system including hardware and software, the overall architecture of the system, and the integration of different parts to create the finished system.

Software Engineering has a system focus because it addresses the whole development process rather than just coding. CASE can be about software, or system. Point tools, e.g. test case generators, are 'software tools'. Lifecycle tools, e.g. Yourdon, SSADM are 'system tools'.

This argument, or equally powerful alternative argument - **5 marks**.

b) A good answer would identify two Software issues from the following set of possibles:

- analyse,
- design (upper CASE early, lowercase late),
- implementation, testing,
- test case generating,
- editor tools

10 marks, with reasons

And two System issues from the list of possibles:

- architecture of distributed topography – Client-server or distributed objects
- Interface definitions for these objects (system functionality)
- Loose integration of distributed objects (to enable cooperation)
- Middleware design to permit interoperation.

10 marks, with reasons

Examiner's Guidance Notes

This was a question about the culture of a software engineer, and asked candidates to be aware of the distinction between 'software' and 'system'. Part (a) asked for a relatively short analysis of 'software' and 'system'. Few candidates gave much analysis, preferring to assert one view or another.

In Part (b), a scenario sketched the application of software components within a system framework and asked candidates to identify with reasons two software elements and two system elements for 20 points, that is, 5 points each. Many candidates identified the system elements, because they were stated plainly in the scenario.

Question 3

3. *a)* Discuss the process of developing a model of a system in terms of object classes with specific attributes and operations, e.g. object class diagrams, from models of how the system will be used, e.g. Use Case diagrams. **(4 marks)**

b) Illustrate your answer from part *a)* with a simple example modelling the high level design of a booking system for a video library, giving both use case diagrams and object class diagrams using UML notation. **(6 marks)**

c) Discuss the various ways using UML that the dynamic behaviour of a system can be modelled. **(6 marks)**

d) Using the same example as in part *b*), model the process of booking a video using either a sequence diagram, state diagram, or an activity diagram. **(4 marks)**

e) Discuss the need for Deployment Diagrams in OO system development. The proposed video booking system is expected to be deployed using the World Wide Web. Provide a simple deployment diagram for this system. **(5 marks)**

Answer Pointers

A good answer covered in part a) one of the various techniques that have been proposed for identifying object classes: Noun/Verb analysis with nouns as objects and verbs as operations; identifying tangible entities (real things) in the application domain and modelling these; behavioural approach looking at the overall behaviour of the proposed system and identifying significant participants, scenario based analysis supported by CRC (class-responsibility-collaborator) cards. Any of these is acceptable as an answer.

Most students simply explained what constitutes a use case and what constitutes a class diagram. Fuller answers would link these the actors and actions in use cases to objects and methods

The example in part b) is to get the students to show that they are capable of applying the bookwork in this answer.

In parts c) and d) the description of various ways to model dynamic behaviour is book work, but the example is required in part c) so that students can show they know how to apply at least one of these.

In the final part of the question, answers requiring bookwork on deployment diagrams were weak and again as in d) few students used the correct notation although most picked up on the fact that deployment diagrams addressed the physical view of the system. Students were less confident with the UML notations relevant to these parts of the question

Examiner's Guidance Notes

This question aligns with the objective that students are able to create models of software data and processes using Object Orientation modelling approaches such as UML. It sets out to assess student's understanding of OO Analysis and Design using UML.

Most students attempting this question did well and demonstrated a good working knowledge of UML book work and ability to use the notations in practice. Under examination conditions, only rough diagrams were expected.

Question 4

4. The following is an outline specification for a project. Select the criteria you would use to determine the life cycle model that this project should follow, and hence make a recommendation about selecting a suitable life cycle model. **(25 marks)**

The Managing Director (MD) of HiJet, a company that cleans drains, uses a PC computer with TV interface to view a video tape of the inside of a suspected pipe. On the MD's PC there is Outlook2000, MS Word, MS Project and the ACT database. These tools capture information (emails), record invoices (Word), record jobs and job progress (ACT). The MD's skill is diagnosis of faults in a pipe.

The firm wants to use its technology to gain control over the progress of a 'job'. The effort and skill must stay with human-intensive diagnosis, but as much as possible of procedural order processing should be automated. An intranet should enable three other screens in three other offices to view aspects of jobs and make changes to show job progression.

Answer Pointers

A good answer would find that the requirements seem plain enough – to integrate existing tools on to the business of tracking a job and ensuring it brings income to the firm. Automation of order-processing implies some form of database, and the intranet should be supported from the database. **10 marks**

The 'elements' of this project can be progressed in sequence, starting with a more complete analysis, the designing the data model, then creating the database, then adding the intranet views on the database.

This powerfully suggests incremental development. **5 marks**

There is no apparent need for an evolutionary approach, or prototyping, because requirements are stable and interfaces are straightforward. There is no difficult decision-taking (it is reserved to the human) and many of the individual elements are already automated. . An alternative might be a Waterfall approach, but only if it emphasised frequent checking-back to ensure requirements are being addressed.

Discussion of alternatives: 10 marks

Examiner's Guidance Notes

This question was popular, but many candidates did not spend enough time to read the description, and offered answers without referring to reasons for their choice and connecting their decisions to elements of the case study.

The question does not offer a breakdown of marks. The candidate should decide how the marks might be allocated, and structure an answer accordingly.

For example, the question asks about factors affecting choice of lifecycle, and then the making of a choice for a lifecycle. This divides the answer into two.

In the first part, there are, perhaps, four or five elements of the case study that might influence choice of life cycle. Each element might earn 3 or 4 marks.

In the second part, a competing set of, say, two or three lifecycle models could be compared against the elements found so far, and a choice made. Three discussions, 4 marks each, or two discussions, 6 marks each. Either is a plausible format of answer.

Question 5

5. *a)* Explain the difference between software validation and software verification during the software life cycle in assuring software quality. **(5 marks)**

b) Various categorisations of software product quality factors have been proposed. Outline ONE of these by explaining the basis for its categories and the factors associated with each category. **(10 marks)**

c) Discuss the types of testing relevant to each of the processes, validation and verification, and indicate which of the quality factors is being assured through each type of testing. **(10 marks)**

Answer Pointers

A good answer gave the standard definitions of validation and verification as per Sommerville or Pressman, following Boehm, e.g. "Are we build the right product?" – validation is the process of establishing that the software product being built is the one required to meet the customers' needs and "Are we building the product right?" – verification is the process of establishing that the software being implemented is correct with respect to its specification and design.

The main differences being that validation requires a dialogue with the intended users and customers of the software; whereas verification requires a dialogue amongst members of the software development team and where formal methods have been employed can be determined formally.

In Part b) the expected answers were either of the two categorisations and sets of factors given in Pressman are McCall et al and the more recent FURPS from Hewlett-Packard.

Sommerville gives a more general account of software product quality in terms of 4 factors:
Process quality, people quality, development technology, and cost, time and schedule.
A few students' answers were based on this.

A few students wrote excellent answers based on the ISO Software Product Quality Standard.

The final part of the question requires students discuss the various types of testing, i.e. unit testing, integration testing, system testing, acceptance testing, regression testing, and to relate these to verification and validation, generally only acceptance testing is a validation activity involving the customer although use cases agreed with the customers/users may drive design and influence subsequent test development indirectly.

The indication of quality factors will depend on which quality factors the student has given in the earlier part of their answer. Few students make this link although some gave very clear tabular answers establishing the link to quality factors.

Examiner's Guidance Notes

Most students were able to distinguish clearly between validation and verification although some lost marks by neglecting to make any link to software quality assurance during the software life cycle.

Software Product Quality categories and factors was very poorly addressed. Few students achieved full marks here; most simply listed various so-called "-abilities" and defined these.

On the final part of the question, students demonstrated a familiarity with various types of testing and were able to link these with validation and verification processes, but few made the link to quality factors.

There were some excellent answers based on the ISO Software Product Quality standard.

Question 6

6. *"Software Product Maintenance is the management of change throughout the whole of the software product life cycle."*

a) Discuss the above statement and explain the necessity for software product maintenance throughout the software product life cycle. **(5 marks)**

b) Outline the primary activities of software product maintenance with respect to its place within the software product life cycle. **(5 marks)**

c) Discuss the various CASE tools that are available to support the activities of software product maintenance. **(5 marks)**

d) Describe ONE such tool in detail indicating how a team of software engineers could use this tool during maintenance and what advantages using the tool will provide. **(5 marks)**

e) A software development organisation has an established practice of software product maintenance using a repository where all versions of their products are stored. They now wish to develop a programme of software reuse. Of what relevance is their software product maintenance practice to their proposed software reuse programme? **(5 marks)**

Answer Pointers

A good answer for (a) contained a discussion about the inevitability of change particularly with respect to software and the need for controlled change in large complex systems. The major activities expected in (b) are those found in descriptions of software configuration management: CM planning, change management, version and release management and system building (from Sommerville), or similar from Pressman: configuration item identification, version control, change control, CM audit, status reporting.

In the CASE tool discussion of part (c), answers which distinguished between comprehensive SCM tools such as ClearCase which support all CM activities, and stand-alone tools such as RCS which support one i.e. version control in this case were expected. The CASE tool described in detail for (d) could be any well known SCM tool, e.g. Microsoft CodeSafe, Rational ClearCase, CVS, Subversion or earlier tools such as RCS or SCCS. Few students provided such an answer; many defined and discussed CASE tools in general.

The final part of the question, (e), expects students to make a connection between the infrastructure to support reuse and that needed for SCM; e.g. defined change processes in place, a software repository, and clear means of identification and versioning of software components, etc. Most answers made this link.

Examiner's Guidance Notes

This question was popular with students. Some made the link between management of change and software configuration management as a key activity of software maintenance immediately, although a number of students gave the received view that software maintenance is an activity that starts when the product is delivered to the customer and failed to address the statement as it stood.

This failure led to rather unsatisfactory answers to part b) although most students understood that maintenance involves repetition of parts of the software lifecycle.

Answers to part d) were rarely focussed on specific CASE tools to support maintenance; and the related part d) suffered from the same problem although there were some excellent answers which went straight to the point and discussed tools for

software configuration management, version control, change tracking, reverse engineering, etc.

Where students attempted the final part, they usually were able to explain what software reuse entailed and how a repository would be useful.