2910210 Software engineering and development

Examiner's report: Zone A

Introduction

The paper has been designed to examine the students' general software engineering knowledge, in particular; software development; software process techniques; software testing; state transition diagram; control flow graph and interface issues.

Of course, the above topics were chosen by the examiner to assess students' knowledge this year and the same topics will not necessarily be featuring in next year's examination. Also, even if the same topics may be featured, the questions could address different aspects or issues. So make sure you have good knowledge of the full spectrum of topics included in the curriculum for this module.

A good understanding of the basic concepts, ideas, methods and techniques is expected of you. Also, the proper use of terminology and notation is particularly important.

Some of the questions are of a straightforward bookwork type that can be answered basically by using material from the subject guide. However, you can also use material from other software engineering educational sources to enhance and supplement your answers. This can demonstrate greater knowledge and deeper understanding, and may lead to the award of higher marks.

The following are not 100 per cent model answers; rather they are indicative statements showing how to answer the questions in a better way.

Question 1

Section a

This is a bookwork type of question and a good answer would give a good description of the three principles:

- Rigour means spelling out soundly and comprehensively each step and each product (including intermediate ones) of the software process, with as much precision as necessary.
- Modularity is especially important in software engineering. Software is composed of modules if, and only if, it is broken up into smaller component units.
- Anticipation of change means taking into account all along the development process that the software must evolve.
- The above is only indicative and you should be more elaborate in your description.

Section b

A good way to answer this section is to give a definition and list the main stages of the spiral and the waterfall models supported by diagrams of both models. A better answer would briefly mention how the models are different approaches born out of the evolution of the SDP development models.

Your answer should include a brief description of the various stages of each model. The example answer below is only indicative and you should elaborate more as this section has a high mark weighting.

Spiral model stages:

- the planning stage – to define resources, time lines and other project-related information
- the risk analysis stage – to assess both technical and management risks
- the engineering stage – to build one or more representations of the application
- the customer evaluation stage – to obtain customer feedback based on the evaluation of the software representation created during the engineering stage and implemented during the install stage.

The waterfall life cycle model has the following stages:

- analysis: this stage involves development of the software system requirements
- design: this stage includes development of the architectural layout of the system, preparing the formats of the main data structures selecting efficient algorithms for the basic units, deciding the software system interface, designing a formal description of the whole system in terms of a pre-selected description language, and eventually building a system prototype
- coding: this stage supposes implementing of the software system using a contemporary programming language chosen with respect to the system requirements
- testing: this stage ensures that all program statements have been verified internally, and all predefined requirements are externally provided to the user
- project maintenance: this is more like a set of additional activities that serve to support the future use of the software after it has been delivered to the client.

Section c

In answering this section you have to show awareness of the main advantages and disadvantages of the spiral and prototyping models, and how they translate into costs and benefits. This information can be found in your subject guide and other texts, but you have to create the discussion.

Question 2

Section a

A good answer to this section would start by stating the obvious advantages of program commenting like being a memory of what has been done for other people to take over etc. One of the main disadvantages is that it is even harder to maintain comments than code, and that non-maintained comments can mislead.

Section b

To answer this question draw on your general reading and choose the important and obvious items that should be included in a header comment. A good one could be: sample calling sequences, a description of the arguments, and a list of subordinate modules.

Section c

The answer to this section should first mention that the program estimates the square root if ‘a’ is positive (also indicate how you reached that). Second, state how each of the categories you mentioned in section b is suitable in this case.

Question 3

Section a

The answer to this section can be very basic and is basically bookwork. No need for elaboration here and brief description of the four testing stages will suffice.

Unit testing: each module is tested in turn and in isolation from the others. Uses white-box techniques exclusively.

Integration testing: unit-tested modules are successively added to an assembled, integrated configuration in a stepwise manner until the entire software is assembled. Fewer white-box and more black-box techniques are used.

Validation testing: the validation criteria from software requirements are applied to the integration-tested software. Uses black-box techniques.

System testing: this testing is concerned with testing the entire computer-based system. It is mainly used to test the interfaces between software, hardware and human components.

Section b

A good answer to this question is to give a good description of the three kinds of system testing.

First, indicate in your answer that system testing ranges over all components of a CBS, not just software.

Recovery is concerned with testing the requirement that the CBS can recover from faults and resume operations within a specified period of time.

Security testing attempts to verify that the protection mechanisms built into the system do prevent intruders from gaining access.

Stress testing exercises, abnormal conditions, typically quantities, frequencies and volumes. The boundaries of normal and abnormal levels are normally captured during the system requirements analysis.

Section c

This section carries a 12-mark weighting and should be answered carefully. Choose a system which you have been familiar with or you have studied, and then describe your testing strategy by relating to the four stages described in section ‘a’. In your description, try to justify how you are going to include the stages in your strategy.

Question 4

Section a

A good answer to this section should be short and to the point as there are only five marks associated with it. The answer should contain the main point that cognitive psychology is about the study of how people learn and the effects of that learning, and certainly talk about the need for producing usable software.

Section b

This section is linked to the previous one, and you can draw on material from the subject guide. A good answer would begin by describing the three terms, and then discuss the main point of treating people's cognitive abilities largely as kinds of information processing. That is, cognitive psychology is the study of how people acquire and use information.

A comprehensive answer may also discuss human recognition (including picture recognition) and recall and mention Miller's 7+-2 rule and its effects on the number of items that should be available at any times.

Section c

To answer this section, choose five other different features of software systems (e.g. undo, automatic save, help, etc.) that you know best, then describe each of them highlighting its use and benefits. Below is an example of the Undo feature.

Undo is a command in most word processors and text editors. It erases the last change done to the text buffer reverting it back to an older state. In some more advanced programs such as graphic processing, undo will negate the last command done to the file being edited. The opposite of undo is redo. Undo is used in many programs and is a helpful feature. On most Windows machines, the Undo command is activated by pressing CTRL+Z.

Question 5

Section a

A good answer to this section should include all the points outlined below including the example.

A state transition diagram depicts states in round-cornered rectangles and the transitions between them as arrows. There is a specific state designated the start state and there maybe some final states. The arrows are typically labelled by the input that causes the transition to occur. The arrows may also be labelled by an output that the transition produces. It is possible for a state to have many transitions to other states, and for the same inputs to lead from a given state to several different states. It is also possible for a transition to lead from a state back to itself.

In software engineering development the states are used to represent the states in which the system can get to, and the transitions to represent options available to a user. For example, we can describe the various options open to a user of a windowing system as transitions and the states could be the state of the screen in an HCI.

Section b

The answer to this section is a state transition diagram for the central heating controller. There are several ways to represent the system, one possibility:

A state is triple: on/off, period (e.g. T12 for between 1 and 2) and setting of change 1 or change 2.

The transitions are triggered by moving from one time period to the next or a setting change.

States:

T12, on, Change 1

T12, on, Change 2

T23, on, Change 1

T23, off, Change 2

T34, on, Change 1

T34, on, Change 2

T41, off, Change 1

T41, off, Change 2.

Question 6

Section a

A good answer to this section will give descriptions for the levels of testing and then describe what 100 per cent coverage at each level means.

Statement coverage: this measure reports whether each executable statement is encountered. In order to achieve 100 per cent statement coverage each statement of the program must be executed at least once.

Path coverage: this measure reports whether each of the possible paths in each function have been followed. A path is a unique sequence of branches from the function entry to the exit. In order to achieve 100 per cent path coverage, every path from the entry node to the exit node of the CFG of the program must be traversed during at least one execution of the program.

Branch coverage: branch coverage (sometimes called decision coverage) measures which possible branches in flow control structures are followed. In order to achieve 100 per cent branch coverage it is necessary to execute each predicate at least twice. In one instance the predicate must evaluate to false and in another it must evaluate to true.

Section b

The answer to this question is a control flow graph:

i → ii
<u>ii</u> → iii and vi
iii → iv and v
<u>iv</u> → v
v → ii
vi → vii and END
vii → viii
viii → ix and x
ix → vi
x → vi

Make sure you use the right notation for control flow graph.

Section c

A good answer for defining a test set for 100 per cent statement coverage will need you to have all the conditions on iii true at least once, so for example x=1 and y=6. This will also suffice to get the condition on vi true. The condition on viii is also satisfied by this condition. So this test case goes to every statement but x. x is unreachable since you can not get to viii with the condition false.