2910210 Software engineering and development

Examiner's report: Zone B

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 more of indicative statements showing how to answer the questions in a better way.

Question 1

Section a

This is a bookwork type of question, and the best way to answer it is to base your answer on material from your subject guide to explain the three software development properties of productivity, timeliness and visibility. You can also use material from other resources to supplement your answer.

Productivity is to the process what efficiency is to the product, and is basically concerned with the efficient use of resources within the software development process. Also, describe what it entails in the context of software engineering where personnel and time are the most important resources.

Timeliness is concerned with delivering the software on time. A software development process is timely if it reaches a successful conclusion within the expected time.

Visibility is crucial for management. An SDP is visible if, and only if, its past, current and future states are clearly and completely documented.

The above is only indicative, and you are expected to give more elaborate descriptions.

Section b

A good way to answer this section is to give a definition and list the main stages of the spiral and the prototyping 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 such as the waterfall.

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

Prototyping model stages:

- requirement gathering
- design
- prototype building
- customer evaluation
- prototype refinement
- product engineering.

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. Mainly used to test the interfaces between software, hardware and human components.

Section b

A good answer to this question is to give a description of top-down and bottom-up followed by an example or a diagram of each one. You can here use the examples and diagrams found in the subject guide.

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 is a discussion type of question which is required to be supported by an example of a psychological experiment. There are many such experiments including how much harder people find it to understand that every domino covers a black and white square on a chessboard then the same task with black squares having pictures of eggs and white of bacon. There is one described in the subject guide involving stamps and envelopes. Your discussion should include explanation and implications.

Section c

A good answer to this question will require you to carefully choose the four software inspection guidelines (the ones you know best and can explain and discuss). Your discussion should include why you think they are useful.

The objectives of an inspection are to uncover errors, ensure correctness, solidify standards and improve management.

The inspection is centred a round a review meeting. The meeting should consist of 3–5 people. The meeting should focus on a product or deliverable. There should be a moderator. The meeting should have a formal agenda and established ground rule. The meeting should be under two hours. There should be a follow up process to insure that action is taken to correct any faults discovered.

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
ii → iii and vi
iii → iv and vi
iv → v
v → iii
vi → vii and END
vii → viii
viii → vi and END

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

Section c:

A good answer with explanation would be:

The main difference between the statement and path coverage is that for path coverage from iii and from vi you need to have tests that involve going through the part that you go through if the IF is false as well as if it is true. With statement coverage you do not need the false one.

So you need a branch from ii: anything with x=1

One that goes from ii straight to vi: anything with x different from 1

From iii to iv: need x=1 to get to iii and also y>3 to get to iv

From iii straight to vi : need x=1 to get to iii and also y<3

From vi to vii: need even x

From vi to END : need x odd

Example test set : (x=1, y=4), (x=1, y=2), (x=2, y=2), (x=3, y=1).