December 2014

98-Comp-A6

Software Engineering

3 Hours Duration

Notes:

1. If doubt exists as to the interpretation of a question, the candidate is urged to submit with the answer paper a clear statement of any assumptions made.

2. No calculators permitted. This is a closed book exam.

3. Answer any six of the ten questions.

4. Any six questions constitute a complete paper. Only the first six questions as they appear in your answer book will be marked.

5. All questions have equal weight.

Marking Scheme

1. 20 marks.
2. 20 marks.
3. (a) 5 marks; (b) 5 marks; (c) 10 marks.
4. (a) 5 marks; (b) 5 marks; (c) 10 marks.
5. (a) 5 marks; (b) 5 marks; (c) 10 marks.
6. (a) 5 marks; (b) 5 marks; (c) 10 marks.
7. (a) 5 marks; (b) 5 marks; (c) 10 marks.
8. (a) 4 marks; (b) 4 marks; (c) 4 marks; (d) 4 marks; (e) 4 marks.
9. (a) 5 marks; (b) 5 marks; (c) 10 marks.
10. 20 marks.

Total mark out of 120.

Question 1. *The Software Design Process.*

Describe the main activities of the software design process and the outputs of these activities. Using a diagram, show possible relationships between the outputs of these activities.

Question 2. *Object-Oriented Software Design.*

Identify possible objects in the following system and develop an object-oriented design for it. Make and state reasonable assumptions about the system when deriving the design.

> An automated ticket-issuing system sells rail tickets. Users select their destination and input a credit card and a personal identification number. The rail ticket is issued and their credit card account charged. When the user presses the start button, a menu display of potential destinations is activated, along with a message to the user to select a destination. Once a destination has been selected, users are requested to input their credit card. Its validity is checked and the user is then requested to input a personal identifier. When the credit transaction has been validated, the ticket is issued.

Question 3. *Software Design.*

(a) Discuss the differences between object-oriented and function-oriented design.

(b) Define the terms *cohesion*, *coupling* and *adaptability*. Explain why maximizing cohesion and minimizing coupling leads to more maintainable systems. How is coupling and software portability related?

(c) A software system is to be developed for a microprocessor-based *Home Security System* (HSS). The system receives input from entry sensors, smoke sensors, temperature sensors and flood sensors. The system is capable of generating alarms, turning on selected lights, and calling owner-specified phone numbers. The system is owner-programmable through a keypad. The owner can set thresholds for the sensors, program phone numbers and set delays for various alarms.

Using a function-oriented approach, derive a design for the HSS described above. Make reasonable assumption and clearly state them.

Question 4. *Embedded Software.*

(a) Define real-time software systems. What is the difference between soft real time systems and hard real time systems?

(b) Identify possible stimuli and the expected responses for an embedded system that controls a home refrigerator.

(c) Use a state-based approach to model the operation of an embedded software system for a voice mail system commonly included in a landline phone. The system should display the number of recorded messages on an LED display and should allow the user to dial in and listen to the recorded messages. Make reasonable assumptions and state them clearly.

Question 5. *Software Testing.*

(a) Explain why testing can only detect the presence of errors, not their absence.

(b) What is regression testing? Explain how the use of automated tests and a testing framework simplifies regression testing.

(c) Give a set of test cases for the following components:

- a sorting routine which sorts arrays of integers.
- a routine which takes a line of text as input and counts the number of non-blank characters in the line.
- a module designed to read in a date expressed using the format YYYY/MM/DD, where YYYY is the year (exactly 4 digits), MM is the month (1 or 2 digits allowed), and DD is the day (1 or 2 digits allowed). Spaces are to be ignored. Thus, valid entries might be 2011/ 1/ 1, 1990 /12/25, or 1250/03/18.

Question 6. *Configuration Management.*

(a) What is configuration management in the context of software systems?

(b) How is configuration management different from version management?

(c) A common problem with system building occurs when physical file names are incorporated in system code and the file structure implied in these names differs from that of the target machine. Write a set of programmer's guidelines that help avoid this and other system building problems that you can think of.

Question 7. *Critical Software Development.*

(a) Describe the three complementary approaches to developing dependable software.

(b) Describe four software engineering techniques that can lead to fault-free software.

(c) Illustrate how the techniques you describe in part (b) above can be used in the design of a software-controlled insulin delivery system that works by using micro-sensors embedded in the patient to measure some blood parameter that is proportional to sugar level and then control a pump to dispense the necessary amounts of insulin via a permanently attached needle.

Question 8. *Software Reliability.*

Suggest appropriate reliability metrics for the following classes of software systems. Give reasons for your choice of metric. Suggest also approximate acceptable values of the system reliability.

(a) A system that monitors patients in a hospital intensive care unit.

(b) A word processor.

(c) An automated vending machine control system.

(d) A system to control braking in a car.

(e) A management report generator.

Question 9. *Component-Based Software.*

(a) What is component-based software engineering?

(b) Why is it important that all component interactions be defined in terms of "requires" and "provides" interfaces?

(c) Design possible interfaces of components that might be used in a system for an emergency control room. The interfaces are for a call-logging component that records calls made, and a vehicle discovery component that, given a postal code and an incident type, finds the nearest suitable vehicle to be dispatched to the incident.

Question 10. *Software Validation.*

Explain why single-threaded (i.e., sequential) software designs are easier to validate than designs that involve multiple threads (i.e., parallel).