University of Glasgow

Thursday, 30th April, 2013
9.30am – 11.30am
(2 hours)

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

CS3X: PROFESSIONAL SOFTWARE DEVELOPMENT 3

Answer all 4 questions

This examination paper is worth a total of 80 marks.

For examinations of at least 2 hours duration, no candidate shall be allowed to leave the examination room within the first hour or the last half-hour of the examination.

INSTRUCTIONS TO INVIGILATORS: Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.

1. Project management

You have been employed as a consultant on a project for a government health department in a large country. The department currently maintains three separate information systems for managing immunisation records, histories of disease outbreaks and nutrition surveys. The department wishes to develop a system to allow researchers to link the different data sets in flexible ways, to support new types of analysis. These analyses could help improve national health policies, by for example, linking forms of malnutrition with occurrences of diseases.

Consider the following schedule of activities for the requirements gathering aspect of the project.

ID	Description	Duration	Dependencies
T1	Specify initial use cases	1	
T2	Define initial domain model	1	
T3	Specify initial system non-functional requirements	2	
M1	Initial requirements specification		T1,T2,T3
T4	Prepare for interview with senior researcher	1	M1
T5	Interview senior researcher	1	T4
T6	Refine requirements specification	2	T5
T7	Build prototype user interface	5	M1
T8	Prepare for system walkthrough with research team	2	T7
T9	Conduct walk through	1	T8
T10	Refine requirements specification	3	T9
T11	Prepare for database coordinator interviews	3	M1
T12	Interview immunisation coordinator	1	T11
T13	Interview disease outbreaks coordinator	1	T11
T14	Interview nutrition surveys coordinator	1	T11
T15	Refine requirements specification	5	T12, T13, T14
T16	Review information privacy legislation	11	T15
T17	Refine requirements specification	2	T16
T18	Investigate system interfaces	5	T15
T19	Refine requirements specification	2	T18
M2	Revised requirements specification		T6,T10,T18,,T19

(a) Describe the advantages and disadvantages of PERT and Gantt charts for visualising project schedules.

[4]

(b) Draw a PERT chart of the tasks described above.

[6]

(c) Calculate the critical path for the project, showing your working on your PERT chart. State the tasks that are on the critical path.

[6]

(d) Suppose that you could recruit an arbitrary number of requirements engineers to work on the requirements gathering and specification tasks.

(i) How would you refactor the schedule to leverage the extra effort available and thus reduce the duration of the project?

[1]

(ii) Given the description of tasks, describe a risk associated with recruiting extra requirements engineers in order to reduce the project duration.

[1]

(iii) What tools and practices would you need to employ to support the revised schedule?

[2]

2. Requirements analysis

(a) There are a number of different methods available for requirements elicitation. Briefly describe three different approaches and note an advantage of each.

[3]

The following scenario is used for parts (b)-(d) of this question.

You are currently working as a business analyst in a company which offers bespoke software systems. Your current client, Mr. Baker, has decided he would like software to manage the storage and distribution of the multitude of recipes he uses in his bakery. He has provided the following information on how he currently manages his recipes.

Mr. Baker currently uses a paper-based system to manage his recipes. He maintains a recipe book which he can add recipes to. A typical recipe has a name, a cooking temperature, a cooking duration (how long the item needs to cook for), an ingredients list, and a method for combining the ingredients. The ingredients can be repeated between recipes, but the quantities might change. Specifically, an ingredient may have three types of quantity, a weight, a number of items or a volume. For example 50ml of milk, 2 lemons, or 100g of butter.

Using this current system introduces a number of problems which Mr Baker hopes would be solved by the new software system:

- Mr. Baker wants to share a recipe with his many friends. Currently he has to locate the recipe and type it into his computer before e-mailing the recipe. To resolve this, Mr Baker would like to make it possible for his friends to view the recipes and add their own. However, he does not want them to be able to remove recipes from his collection or edit his existing recipes.
- Mr. Baker currently has to traverse his entire collection to locate a desired recipe. This is not ideal and Mr. Baker would like to be able to search for recipes based on ingredients (e.g. lemon or chocolate) and also on the recipe name.
- The current method of editing a recipe includes locating the recipe and re-writing it. Mr Baker would like the new system to deal with changing a recipe more effectively.

(b) Identify the actors of the system. Draw a use case diagram incorporating the most important facilities of the recipe collection management software. Provide a short description (one sentence maximum) of each use case which has been identified. Note any assumptions which would need to be examined by further requirements gathering.

[9]

(c) Draw a domain model as a class diagram, outlining the key classes, their attributes, and relationships that would be needed to support the activities identified in your class diagram. You do not need to show the methods of your classes or the types or visibility of the attributes which you identify.

[5]

(d) As a result of further requirements gathering, you have discovered that Mr. Baker would like to have the facility to add reviews of his recipes. Other users of the system should also be able to do this. A review will consist of a comment and a rating out of five stars.

How would your **class diagram** be adapted to incorporate the new functionality? You need not re-draw your diagram.

[3]

3. Component-based software engineering.

(a) What is a software *component*?

[2]

(b) Give three benefits of component-based software engineering.

[3]

The following scenario is used for parts (c) to (e).

In an online messaging system, each node has a unique id. If a *caller* node wants to communicate with a *receiver* node, then the caller looks up the receiver id r on a central *addressing server*. Once the caller knows r, it can send messages directly to the receiver, and listen for replies coming back from that receiver. (Assume the receiver node does not need to access the central addressing server to reply, since the caller id is embedded in the caller's messages.)

(c) Draw a UML component diagram for this online messaging system.

[5]

(d) Which two *architectural design patterns* are exhibited by the component interactions in this system?

[2]

(e) Now suppose the address server is reimplemented as a load-balanced farm of N servers. Explain (perhaps with a diagram) how the new address server might be deployed so as to minimise disruption to the existing system.

[4]

(f) Name two different data items that should be stored as *component metadata*. Justify why each item is important for component management or execution.

[4]

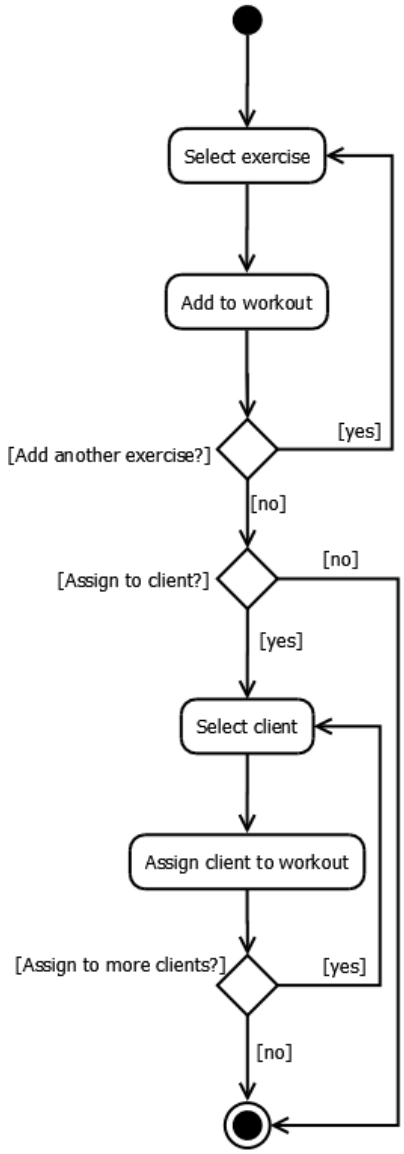
4. Testing and evolution

(a) What four pieces of information do you need about a non-functional requirement in order to develop an acceptance test?

[4]

(b) Consider the following scenario and then answer the question below.

You work for a company which is developing a software system for a personal trainer who wants software to keep track of his clients - their contact details, scheduled sessions, and workouts consisting of a number of exercises which can be assigned to the client. A specification was agreed with the client, and the system has been built. The process has now reached the stage of acceptance testing. You have been assigned a use case "create workout". Creating a workout allows the trainer to combine individual exercises in the system to create a total workout by selecting from the available exercises. This workout can optionally be assigned to one or more clients by selecting a client from those available. The use case is described in the activity diagram shown below:



Give a brief description of the acceptance tests you would write to demonstrate that the implementation has achieved the functionality detailed in the activity diagram.

[5]

(c) Refactoring should be part of the software development process, but when should it occur and why?

[2]

(d) Briefly describe the refactoring process.

[3]

(e) A shift cipher takes each letter in a plaintext and shifts it a specified number of letters along the alphabet. If the letter then gets beyond z, it starts at the beginning of the alphabet again. For example if you had a shift cipher which shifted all letters by 3, then the plaintext alphabet and the cipher alphabet are as shown in the diagram below:

Plain Alphabet

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Cipher Alphabet

D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

The following method, "setAlphabet" has been written to aid an implementation of a shift cipher. The method has a parameter `p` which represents the number of places to shift the cipher. It then initializes two alphabet arrays, one to hold the plain alphabet, one to hold the cipher alphabet. The cipher alphabet array is then shifted by `p` letters.

```
private char [] alphabet=new char[26];
private char [] cipherAlphabet=new char[26];
private char firstLetter='a';

private void setAlphabets(int p)
 {
  for(int i=0;i<26;i++)
  {alphabet[i]=(char)(i+firstLetter);}

  for(int i=0;i<26;i++)
  {cipherAlphabet[i]=(char)(i+firstLetter);}

  for(int i=0;i<26;i++)
  {int value=cipherAlphabet[i]+p; if(value>'z') value-=26;
  cipherAlphabet[i]=(char)(value);}
 }
```

The method requires refactoring. Briefly describe three issues with the code and steps you could take to remedy them. You do not need to re-write the code.

[6]

END OF QUESTION PAPER