BCS IMIS HIGHER DIPLOMA QUALIFICATIONS

Contemporary Issues in Systems Design (H3)

Wednesday 4th December 2013 10:00hrs – 13:00hrs

DURATION: 3 HOURS

Answers

Part A: Answer all of the following 8 questions (5 marks each)

The purpose of these questions is to establish that the students understand the basic ideas that underpin the course. The answers should be largely descriptive and quite short.

1. From the perspective of the end-user identify THREE types of "quality" problem that may arise during systems development.

Quality problems affect the nature of what is achieved. The topic is discussed pp Page 52-53 in the course text where the following problems are identified: The wrong problem is identified, the context is neglected, incorrect requirements analysis, and project carried out for the wrong reason, etc. Three of these should be identified. 2 marks for each to a maximum of 5.

2. From the perspective of the system developer identify THREE types of "productivity" problem that may arise during systems development.

Productivity problems relate to the rate of a progress of a project. The topic is discussed pp Page 55-56 in the course text where the following problems are identified: Requirements drift, external events, poor project management, implementation not feasible, etc. Three of these should be identified. 2 marks for each to a maximum of 5.

3. Explain a MAIN difference between the terms "method" and "methodology" as used in information systems design.

The key point that must be made is that a methodology is a framework that describes an overall approach to software development (1); typically comprising a programming paradigm (e.g. object orientation) (1) a set of techniques and notation (e.g. UML) that support the approach (1), and a lifecycle model (e.g. spiral and incremental) with phases and structure (1). We might say the USDP is a methodology. A method is an instantiation of the methodology in a given situation i.e. for a specific project (1).

4. How does the prototyping approach to systems development differ from an approach based on incremental development?

Prototyping in not necessarily concerned with delivering a working system (1) whereas an incremental approach delivers a working system in successive increments (1). In an incremental development method such as the Unified Software Development Process (1) an increment may be any life cycle product and does not necessarily resulting in a change to the user interface (1). In prototyping the emphasis is almost exclusively on the interface to the system (1).

5. A "pattern" in the context of object oriented design consists of THREE elements. Describe each of the THREE elements.

A pattern is usually described as an abstract solution to a commonly occurring problem in a given context. We usually consider a pattern to comprise three elements: a context in which a given problem occurs repeatedly, a set of forces that influence or constrain the possible solutions and a software configuration that allows these forces to be resolved. 2 marks per element to a maximum of 5.

6. How do "light" development methods (such as eXtreme Programming) differ from model-centric approaches such as the Unified Software Development Process (USDP)?

"Light" is taken to mean several things, in particular: The method is 'agile' – it emphasises minimum documentation and doing no more than is necessary to achieve the desired business result (1). The method is adaptable, and can be readily tailored to suit the needs of a project (1). XP certainly aims to meet these aims. It is therefore suited to projects with small teams in envronments characterised by changing requirements (1). USDP is more formal – requires lots of documentation and is better suited to larger-scale projects (1). The underlying philosophy of model-centric methods like USDP is that there is value in developing a knowledge hub for the software development lifecycle based on models that are both concise and expressive across the development process (1) – the light approaches do not preserve models in this way.

7. What role does an "abstract" class play in the design of reusable software?

An abstract class is a class that can have no instances (1); a superclass that acts only as a generalised template for its instantiated classes (1). They are therefore used as the basis for creating specialised classes to be used in specific situations having inherited generic behaviour (2) – the essence of reuse (1).

8. For each of the following THREE terms, "syntactic correctness", "consistency" and "completeness" explain their function in Unified Modelling Language (UML) diagrams.

Syntactic correctness is concerned with using the notation (e.g. UML) correctly, consistency relates to producing a set of models or diagrams that are consistent with each other and completeness refers to producing models that are completely defined. 2 marks for each question to a maximum of 5 marks.

Part B: Answer 3 of the following 5 questions (20 marks each)

Question B9

a) Describe, with suitable examples, what is meant by the following class stereotypes used in object oriented design:

i. Boundary Classes. (4 marks)

Boundaries are objects that interface with system actors: UserInterface, DataBaseGateway, ServerProxy, etc. Most significantly boundary classes include web pages, windows forms and so on – the part of the system with which the user interacts.

ii. Entity Classes. (4 marks)

Entities are objects representing system data: Customer, Product, and Transaction. Most significantly entity classes will appear in the domain model or class diagram and typically map to tables in the database.

iii. Control Classes. (4 marks)

Controls are objects that mediate between boundaries and entities. They orchestrate the execution of commands coming from the boundary by interacting with entity and boundary objects. Controls often correspond to use cases and map to use case controllers in the design model. Sometimes control classes are used to implement functionality that does not "fit" into either of the other two types of class.

b) Explain with the use of a suitable example how robustness analysis can be applied to produce an analysis class diagram for a Use Case. (8 marks)

Robustness analysis is concerned with the transition from analysis into design (1). The technique is described in pp. 197 – 201 of the course text. It is concerned with identifying a set of classes that are robust enough to satisfy all the requirements of a use case (1). The basic process involves deriving a sequence diagram for each Use case (1) and then taking the classes identified in the sequence diagram into a class diagram (1). To explain the process properly the examinee will have to make use of a suitable example (4).

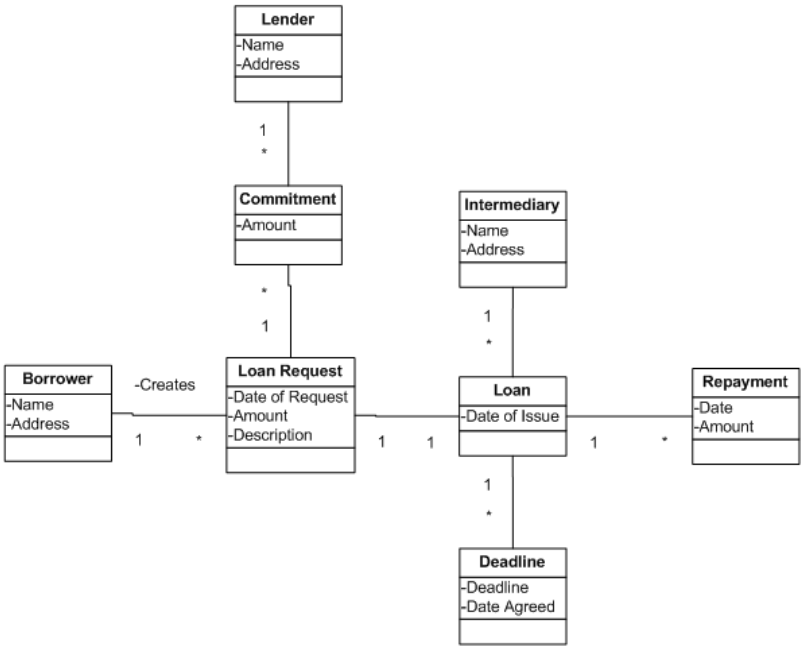
Question B10

a) Identify and briefly describe FOUR Use Cases for the Micro Loans system. (8 marks)

The most important Use Cases will be: Request Loan, Offer Money for Loan, Make Payment, Agree Deadline. Others are possible. 2 marks for each appropriate Use Case.

b) Create a class diagram for the system from the above information. Include any attributes that you think may be needed. (12 marks)

The following would be worth top marks: 6 marks for identifying the correct classes, 3 for the attributes and 3 for the relationships.



Question B11

a) Explain the main differences between the "hard" and "soft" approaches to systems development. (6 marks)

Hard systems (HS) involves simulations, often using computers and the techniques used in operations research. Hard systems look at the "How?" meaning, how to best achieve and test the selected option of development and analysis (1). Hard systems methodologies are useful for problems that can justifiably be quantified (1). However, it cannot easily take into account unquantifiable variables (opinions, culture, politics, etc), and may treat people as being passive, rather than having complex motivations (1). In contrast soft systems methodologies (SSM) are used to tackle systems that cannot easily be quantified, especially those involving people interacting with each other or with "systems". (1) Useful for understanding motivations, viewpoints, and interactions but, naturally, it doesn't give quantified answers (1). Soft systems looks at the "What?" of the system (1).

b) Identify five general advantages that are claimed for using a methodology to guide systems development. (7 marks)

Among the main advantages usually claimed are these:

- *Better quality system product*
- *More complete satisfaction of user requirements*
- *Greater management control*
- *Better communication among developers and with users*
- *Capture of know-how and its dissemination through the organization*

2 marks for each advantage identified to a max. of 7 marks

c) Identify four disadvantages of using an inappropriate methodology during systems development. (7 marks)

This can lead to solving the wrong problem or developing a wonderful system that nobody needs. Disproportionate cost and delay if the methodology is too complex; quality problems if its techniques do not provide adequate models of the application; user rejection if it does not encourage sufficient participation.

2 marks for each disadvantage identified to a max. of 7 marks

Question B12

The following systems development activities are embodied in the Unified Software Development Process. Identify the Unified Modelling Language (UML) diagrams that are used during each of these activities and explain the techniques that are used within that activity.

a) Requirements capture and modelling. (5 marks)

The techniques applied during this activity include Use Case modelling, architectural modelling and prototyping (2). At the end of the activity we should deliver a use case model, initial architecture or domain model and prototype interfaces (1). During the activity we would make use of Use Case Diagrams and Package Diagrams (2).

b) System Architecture and Design. (5 marks)

The techniques applied during this activity include Deployment Modelling, Component Modelling, Package Modelling, Architectural Modelling and Design Patterns (3). During the activity we would make use of Package Diagrams, Component Diagrams, Deployment Diagrams and Class Diagrams (2).

c) Interface Design. (5 marks)

The techniques applied during this activity include Class and Object Modelling, Interaction Modelling, State Modelling, Prototyping, Design Patterns (2). During the activity we would make use of class diagrams, object diagrams, sequence diagrams, state machines and package diagrams (3).

d) Data Management Design. (5 marks)

The techniques applied during this activity include Class and Object Modelling, Interaction Modelling, State Modelling, Prototyping, Design Patterns (2). During the activity we would make use

of class diagrams, object diagrams, sequence diagrams, state machines and package diagrams (
key output is likely to be a normalised database design specification (1).

Question B13

a) Explain what is meant by "architecture" in the context of information systems design. (4 marks)

The word "architecture"in this context refers to the high level structures of a software system (1). It can be defined as the set of structures needed to reason about the software system (1), which comprise the software elements, the relations between them, and the properties of both elements and relations (1). These are often represented graphically in a modelling language such as the UML (1).

b) Describe the following views of architecture used in the Unified Software Development Process (USDP). Identify the UML diagrams associated with each view.

i. Logical view (3 marks).

The logical view is concerned with the functionality that the system provides to end-users (1). UML Diagrams used to represent the logical view include Class diagram, Communication diagram, Sequence diagram (2).

ii. Development view (3 marks).

The development view illustrates a system from a programmer's perspective and is concerned with software management (1). This view is also known as the implementation view. It uses the UML Component diagram to describe system components. UML Diagrams used to represent the development view include the Package diagram (2).

iii. Process view (3 marks).

The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behaviour of the system (1). The process view addresses concurrency, distribution, integrators, performance, and scalability, etc. UML Diagrams to represent process view include the Activity diagram(2).

iv. Physical view (3 marks)

The physical view depicts the system from a system engineer's point-of-view (1). It is concerned with the topology of software components on the physical layer, as well as the physical connections between these components. This view is also known as the deployment view. UML Diagrams used to represent physical view include the Deployment diagram (2).

v. Scenarios (4 marks).

The description of an architecture is illustrated using a small set of use cases, or scenarios which become a fifth view (1). The scenarios describe sequences of interactions between objects, and between processes (1). They are used to identify architectural elements and to illustrate and validate the architecture design (1). They also serve as a starting point for tests of an architecture prototype (1). This view is also known as use case view.

A Questions	*LO1*	*LO2*	*LO3*	*LO4*	*LO5*
1	√				
2	√				
3		√			
4		√			
5				√	√
6		√			
7					√
8			√		
B Questions					
1			√	√	
2			√		
3		√			
4	√		√		
5				√	