Analysis, Design and Implementation

December 2015

Sample Exam Marking Scheme

This marking scheme has been prepared as a **guide only** to markers. This is not a set of model answers, or the exclusive answers to the questions, and there will frequently be alternative responses which will provide a valid answer. Markers are advised that, unless a question specifies that an answer be provided in a particular form, then an answer that is correct (factually or in practical terms) **must** be given the available marks.

If there is doubt as to the correctness of an answer, the relevant NCC Education materials should be the first authority.

Throughout the marking, please credit any valid alternative point.

Where markers award half marks in any part of a question, they should ensure that the total mark recorded for the question is rounded up to a whole mark.

Marker's comments:

Moderator's comments:

Mark:	**Moderated mark:**	**Final mark:**

Penalties applied for academic malpractice:

Answer ALL questions

Marks

Question 1

a) 'Software Development Techniques are first and foremost about communication'. Discuss this statement **and** say whether you think it is valid or misleading. **6**

While many software development techniques are documentation heavy (1 mark), their key role is to aid in communication between clients and developers (1 mark) and between developers and other developers (1 mark). If they do not aid in this regard, they are being misapplied (1 mark). The more people involved in a project, the harder communication without formal tools becomes (1 mark) and subsequently, this would be a valid claim to make (1 mark).

b) Explain what is meant by the term *scope* in relation to software engineering **and** provide ONE (1) example of a scenario in which scope would be relevant. **4**
Scope is the degree to which existing systems (1 mark) fall within our remit as software engineers (1 mark). Things that are outside our scope are things we cannot change (1 mark). For example, scope would be relevant in developing a login system where we cannot change the way in which a database is structured (1 mark).

Total: 10 Marks

Question 2

a) Explain the term *problem domain* **and** briefly explain the relevance of a problem domain to the design phase of object-orientated analysis and design. **4**
The problem domain defines the subject specialist information around a system (1 mark), especially that information which is relevant to our system (1 mark). Generally, the problem domain will be a subset of the overall context of a system (1 mark), and it helps define for us the jargon and assumptions of the systems we will be analysing (1 mark).

b) Briefly explain the SSADM and UML techniques and **state** ONE (1) diagram that belongs to **each** technique. **6**
SSADM is a data-oriented technique (1 mark) which emphasises the flow of information through a system (1 mark), whereas UML is an object oriented technique (1 mark) which emphasises the co-location of data and the operations that act upon it (1 mark). Data Flow Diagrams are an SSADM technique (1 mark) whereas class diagrams are a UML technique (1 mark).

Total: 10 Marks

Marks

Question 3

a) Object-orientation is sometimes seen as a solution to monolithic software design. Briefly discuss what is meant by the term *monolithic* **and** explain how object-orientation can resolve the problems with monolithic software. **6**

Monolithic programs are those which are large and unwieldy (1 mark) and usually built with a limited amount of modularity (1 mark). By adopting object orientation, modularity is increased (1 mark) and the internal functions of a system can be effectively subdivided (1 mark), making them easier to maintain (1 mark) and easier to build (1 mark).

b) Briefly explain the term *iterative* in relation to modern software analysis and design **and** explain how iterative processes contrasts to traditional systems such as the Waterfall Model. **4**

Analysis and design processes are iterative – we rarely get things right the first time (1 mark) and so we must continually return to earlier stages of the process (1 mark). The Waterfall Model posited a stately procession from stage to stage (1 mark), which did not reflect the realities of software development (1 mark).

Total: 10 Marks

Question 4

a) Explain how Natural Language Analysis (NLA) might be used in the construction of a class diagram **and** outline how it is possible to evaluate potential attributes for inclusion. **6**

NLA is a tool for performing analysis on texts (1 mark) with the intention of extracting useful insight into a software system (1 mark). It works by identifying nouns, verbs and adjectives (1 mark) and assigning them candidate status as classes, methods and attributes (1 mark). With attributes, we might look for redundancy (1 mark) or elements that suggest they would fall outside our project scope (1 mark).

b) Define the term *prototyping* **and** briefly describe TWO (2) different techniques associated with the term. **4**

Prototyping is the process of building a limited proof of concept system (1 mark) with placeholder functionality that shows a client how a system would function (1 mark). Evolutionary prototyping then takes this proof of concept and develops it into the full application (1 mark). Throwaway prototyping discards the proof of concept and writes a new version of the software on the basis of insights gleaned from the prototype (1 mark).

Total: 10 Marks

Marks

Question 5

a) Outline the role of the use-case diagram **and** provide ONE (1) example of this diagram for interacting with a simple on-off light-switch. **4**

The use-case diagram is used to highlight the possible interactions (1 mark) that an actor may have with a system (1 mark). Two marks for an example, which should show the actor (1 mark) and 'switch on' and 'switch off' interactions (1 mark).

b) Explain the role fulfilled by a UML activity diagram **and** describe how it is used in the construction of object oriented computer code. **6**

The activity diagram outlines formal logic (1 mark), taking a process and encoding it as loops (1 marks) and branches (1 mark). It does so with reference to existing classes (1 mark), which means that it largely represents a diagrammatic pseudocode (1 mark) that will be converted into platform-specific implementation (1 mark).

Total: 10 Marks

Question 6

a) Explain the relationship between the factory and abstract factory design patterns. Outline ONE (1) scenario in which the combination of these patterns might be an appropriate solution. **6**

The factory is an object which creates other objects (1 mark) by taking in configuration details (1 mark) and using polymorphism to provide the correctly set up instances (1 mark). The abstract factory is a factory for factories (1 mark). We might use these in combination to create a skinnable interface (1 mark), where the abstract factory generates the right kind of theme factory and theme factory generates the right kind of widget (1 mark).

b) Explain the role of the three parts of the Model View Controller (MVC) design pattern **and** provide ONE (1) advantage of using the pattern in software development. **4**

The MVC is made up of the model, which is business logic (1 mark), the view which is presentation (1 mark) and the controller which is user input (1 mark). The advantage of the pattern is that it allows for full decoupling of responsibilities in a user facing application (1 mark).

Total: 10 Marks

Marks

Question 7

a) Provide TWO (2) advantages and TWO (2) disadvantages of using design patterns within object-oriented programs. **4**

Design patterns represent good solutions (1 mark) that often simplify complex object relationships (1 mark). However, they can be complex to implement (1 mark) and if over-used can lead to designs becoming bloated and unwieldy (1 mark).

b) Briefly explain what is meant by an architecture system measure **and** provide FOUR examples of traits that would fall into this category. **6**

An architectural measure relates to the way in which a system was coded (1 mark) and represents a view of how well this implementation has worked (1 mark). Examples of traits would be maintainability (1 mark), portability (1 mark), reusability (1 mark) and testability (1 mark).

Total: 10 Marks

Question 8

a) Briefly discuss what is meant by a benchmarking harness **and** explain why such a system would be used in assessing the quality of a software system. **6**

Benchmarking is the process of locating performance issues within code (1 mark) by comparing how long different parts of the system take to execute (1 mark). To do this, it's necessary to take modules of code and test them in isolation (1 mark), which can be automated through the use of a benchmarking harness (1 mark) for each platform to which the software is to be deployed (1 mark). This allows for the developer to identify opportunities for optimisation (1 mark).

b) Define the terms *content coupling* and *callback coupling*. You should say which term is worse in relation to software development **and** justify your answer. **4**

Content coupling is seen when an object makes uses of another object's internal variables (1 mark). Callback coupling allows an object to maintain a list of interested 'listeners' which is populated at runtime (1 mark). Content coupling is worse (1 mark) because it is a structural aspect which cannot be changed at runtime (1 mark).

Total: 10 Marks

Marks

Question 9

a) Consider the class definition below. Ignoring the potential impact of change, outline three possible refactorings that could be sensibly performed: **6**

```
public class Person {
  public int gender;
  public String n;

  public Person(String name, int g) {
      n = name;
      gender = g;
   }

   public boolean isMale() {
      return gender == 1;
   }

   public Boolean isFemale() {
      Return gender == 2;
   }
}
```

The maximum number of marks awarded for this question is 6. Award up to 2 marks for each refactoring. Appropriate refactorings include: changing gender to be an enum; breaking name into a 'forename' and 'surname'; and handling the gender comparison via the definition of a static final variable.

Note: 1 mark can be awarded for partially correct solutions

b) 'Refactoring must be respectful of other developers'. **4**

Discuss this statement **and** say whether you think it is valid or misleading with reference to the concept of *impact of change*.
The impact of change defines the extent to which an alteration will impact on other systems (1 mark), and we may not be responsible for those systems in a multi-developer team (1 mark). When we make changes, it is disrespectful to ignore the impact they may have in systems we do not control (1 mark), and so this is a valid statement of good developer practice (1 mark).

Total: 10 Marks

Marks

Question 10

a) Explain how *Test Driven Development* works **and** outline its importance in iterative maintenance. You should also the main benefit that accrues from its use. **6**

Test Driven Development (TDD) works by defining the failure criteria (1 mark) before we write the code (1 mark) and then testing each of these criteria every time we change anything in a program (1 mark). Since we often introduce bugs as part of fixing bugs (1 mark), TDD in an iterative development system will tell us instantly when we have broken existing code (1 mark), saving developer time and effort (1 mark).

b) Provide the code required to implement the following class diagram: **4**

Book
+ Title: String + ISBN: String # year: String
+ setupBook (t: String, I : String, y: String) : void

The following implementation is in Java, but any other language (or pseudocode) would be appropriate:

```
public class Book {
  public String title;
  public String: ISBN;
  protected String year;

  public setupBook (String t, String I, String y) {
    title = t;
    ISBN = I;
    year = y;
  }
}
```

Marks should be allocated as:

The maximum number of marks awarded to this question is 4. Award 1 mark for each attribute with the correct visibility up to a maximum of 2 marks. Award 1 mark for each constructor up to a maximum of 2 marks.

Total: 10 Marks

End of Examination Paper

Marks

Learning Outcomes matrix

Question	Learning Outcomes assessed	Marker can differentiate between varying levels of achievement
1	3	Yes
2	1, 2	Yes
3	1, 2	Yes
4	1, 6	Yes
5	2, 4, 5	Yes
6	1, 4, 5	Yes
7	2, 3, 4, 5	Yes
8	2, 5, 6	Yes
9	2, 5, 6, 7	Yes
10	3, 5, 6	Yes

Grade descriptors

Learning Outcome	Pass	Merit	Distinction
Understand the seamless transition from OO Analysis to OO Design.	Demonstrate adequate level of understanding	Demonstrate robust level of understanding	Demonstrate highly comprehensive level of understanding
Understand how to convert OO analysis and design models to code	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard
Understand the quality attributes associated with an OO development	Demonstrate adequate understanding of quality attributes	Demonstrate robust understanding of quality attributes	Demonstrate highly comprehensive understanding of quality attributes
Understand the concept of maintenance within an OO development environment	Demonstrate adequate level of understanding	Demonstrate robust level of understanding	Demonstrate highly comprehensive level of understanding
Be able to produce OO analysis and design models using a case tool	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard
Be able to convert OO analysis and design models to code using an appropriate IDE	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard
Be able to refactor an OO programme to improve quality	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard