**COIT20258 Software Engineering**

**Term 2 2020**

**Take Home Test**

**Unit Coordinator, Lecturer, Tutor**

**Dennis Jarvis**

**Submitted By:**

**Bishal Budhathoki**

Table of Contents

[Question 1 a. 3](#_Toc53730393)

[Question 1 b. 3](#_Toc53730394)

[Question 1 c. 3](#_Toc53730395)

[Question 2 a. 4](#_Toc53730396)

[Question 2 b. 4](#_Toc53730397)

[Question 3 a. 5](#_Toc53730398)

[Question 3 b. 6](#_Toc53730399)

[Question 4 a. 7](#_Toc53730400)

[Question 4 b. 8](#_Toc53730401)

[Question 5 a. 8](#_Toc53730402)

[Question 5 b. 8](#_Toc53730403)

# Question 1 a.

For the given required functionalities, I consider the following Software Development Methodologies:

1. The Waterfall Model
2. Rapid Application Development
3. Agile Method

The Waterfall Model: It is a plan driven model and has a separate and specific phase of specification as given in the question along with software design, implementation, testing and maintenance.

Rapid Application Development (RAD): Based on prototypes, here the working version of the system application is developed and continuously iterated on that prototype and both the developer and the customer work together.

Agile Method: Like RAD, iterative, but agile works at one feature at a time and the customer checks the feature and after validating the feature next feature is started to develop. Sprint is used to make sure that each feature is developed, tested and presented to customer and follow the planned schedule to complete the project.

# Question 1 b.

Staging the work for each methodology are different for every one of them but they have:

1. **Specification:** It defines **what the client needs.** Even though the required functionalities are provided, it is better to consulate with the system users to understand the system's services, constraints, and goals. These along with the given required functionalities can be used and defined in detail and use as a system specification.
2. **Design and implementation:** It define **how to meet those needs**. Each feature is studied and separated as hardware or software based on architecture of the system. Relation and abstraction of system software is done and then set of programs is written on them.
3. **Validation:** It defines if **it meets the client's needs**. Here each unit is verified and validate to see if it meets the requirement and specification. Here the client can also be involved.
4. **Software evolution:** It define if **it adapts to changes in client needs.** For this, if any changes are needed as per the client or if it does not meet the requirement and specification, then the changes are made and moved to new requirements. For waterfall model, when the above stages are reached then the changes cannot be made until this stage is reached.

# Question 1 c.

Here I choose Agile Methodology. For this, the additional information that is needed are:

1. How big is the system that is being developed?

This method is suitable for small co-located team who can communicate informally. This is not possible for huge development team.

1. Type of system being developed

Plan driven system is best but if requires a lot of analysis before putting in implementation requires detailed design.

1. Expected System lifetime

May require more design documentation even though the agile developer thinks of it as not much of the use for long term.

1. Subject to external regulation

If external regulator needs to be involved for approval the extra documentation in detail is required as part of the system safety case.

# Question 2 a.

The professional responsibilities of a software developer with respect to the testing and documentation of code that the individual develops are:

1. Communicating with the client to understand the requirements of the system product.
2. Software requirement review and test cases/scenarios preparation.
3. Analysing the test results on errors, bugs, usability.
4. Doing test on usability of the software.
5. Preparing reports related to software testing of own written code or used/modified code and then reporting to the design team.
6. Also participating in design reviews
7. Design tests to minimise risk.
8. Perform manual and automated test
9. Write bug reports, troubleshoot and solve issue
10. Provide quality assurance.

# Question 2 b.

Software testing is a process to determine the functionality of a software system to figure out whether the developed system software has satisfied the specified requirement or not and if any bugs are present or not for the quality assurance of the product.

Developer would not be involved in the following aspects of testing and documentation:

1. Release testing
2. User Testing

This is the stage where user or customer gives input and advise on system testing. There are 3 types of user testing namely:

1. Alpha testing: User of the software works closely with developers at developers’ site.
2. Beta Testing: Available to the public and the users report them with issues and bugs.
3. Acceptance testing: Customer tests the system to find the intended purpose of the software and is ready for the deployment in customer environment.

The reason behind the developer not being involved are as follow:

**Issue sprinting from one to another:** This means that developer moving from one functionality to another and working to make second functionality to work brings bugs or even logical error and even if there is no proper understanding in another functionality then the chance of being the system software a failure a some point is evident.

**Test the part of a code or a function at a time:** This means developers tests the functional after it gets completed and no other testing is done like testing after 2 or more functionality is merged. So, for this a separate team is required.

**Less experienced as tester:** This is also a reason as developers being more oriented towards making a software and having less work on testing. On the other hand, testers can find bugs and issues that the developers can miss out very easily and have better understanding of the testing tools as they are used to it.

**Not admitting the flaw in own work:** When an issue or bug is found, developers can hide their fault and when later can blame other or on removing the bug could be an issue for them, developers can not always do the testing of their own work.

# Question 3 a.

Common reuse practises that are employed in the requirements, design and implementation activities of a project are:

1. System reuse
2. Application reuse
3. Component reuse
4. Object and function reuse

**System reuse:** Reuse of complete system that may include several application programs.

**Application reuse:** Without making changes or developing application families from another application, an application can be reused.

**Component reuse:** From sub-system to single objects, component of an application may be reused.

**Object and function reuse:** Single well-defined object can be reused if has been implemented in small scale software components.

Circumstances each practice can be employed are:

**System reuse:** If entire application system is to be reused, adding or modifying code, systems own configuration interface and need of going commercial then system reuse can be done.

**Application reuse:** If application is to be reused but not the complete application and if needed make necessary changes then application reuse can be implemented.

**Component reuse:** Components are collection of objects and its classes that combined to get related functions and services. If the component is to be modified to adapt new environment or extend the code functionality then component level reuse.

**Object reuse:** When there is a need to use the object from a library this is implemented and for implementation need to find correct library, find if objects and methods provide the functionality that is needed.

# Question 3 b.

DataSource class in CaseStudy 2 does represent a good use of inheritance. But their improvements are required. They are:

1. Remove unwanted semicolumn like

// Connect to the data source. Override if source is a database

public void connect() {

};

And,

// Disconnect from the data source. Override if source is a database

public void disconnect() {

};

1. Although Player object is created, there is no datatype Player. Even though there is Player.java file in model of the project, it has not be used or defined to create Player class. Same goes for Room.java and its class file.
2. The load method given is abstract which will be used by child classUse in child class is shown below:-

public DataBaseSource extends DataSource {

/\*

Protected table field can be accessed

Else only public fields

\*/

@Override

public void load() {

}

// implementing connect and disconnect method

@Override

public void connect() {

super.connect();

}

@Override

public void disconnect() {

super.disconnect();

}

public static void main(String[] args) { //main method

}

}

# Question 4 a.

The 3 key issues that the version control software can help to address and solve are:

1. Storage Management
2. Code Overriding
3. Reverting the fault

**Storage Management:** Code is backed up on the server of every developer. This can cause more storage capacity than required. So, code is backed up on the main server but utilises delta to for each version thus saving space.

**Code Overriding:** If code is pushed to the main server by every developer working on the same project, code overriding is the huge issue. Now version control software makes a branch for every developer, everyone works on that branch and when it is combined in the main server, only the changed code is updated.

**Reverting the fault:** When main server is updated with a wrong or buggy code or the code in the wrong server, reverting the changes done is issue without version control. Version control software provides this option to revert to the previous as the code is backed up.

# Question 4 b.

With the given statement I do not fully agree with it. Only class diagram is not enough to understand the complete context of the program. It can give an individual just a brief idea of various class and the structure of the system software and methods and mostly class diagram does not contain complete methods, and schemas that are used. For the better understanding all various diagrams like, sequence diagram, use case diagram activity diagram and more are needed.

Class diagram is only for the static view of the application and is considered from the top-level view only. Every function defined as module is classified as class and have attributes and methods used. Relationship between individual classes are defined along with their association. So other multiplicities are possible. Then there are behavioural models, data driven models, event driven model and more for the better understanding of the system software.

Class diagram has evolved to define/design code modules and its related objects and various other components. It has been like a blueprint for the system software.

Application Developers and their team member have high stake in understanding and implementation of an application. Here, they create class diagram initially then further diagram is made else there would not be other diagram existing for the application development.

# Question 5 a.

While using the inheritance in more extent design becomes unwieldy and inflexible and if needed to add a function then huge code is to be done and requirement might not be fulfilled. How super class will become is also not sure and outside of control and superclass can become super heavy as each class keep depending on after another. If try to change superclass for the sub class use, risk of braking other user subclass increases.

Now composition brings more flexibility making multiple classes functionality merge into one. It can be used without any issue and if a component is to be created only when needed then such delays can be made. Class can also be made that can change the component dynamically. This can not be done in inheritance. Composition then reduces the number of classes to use.

Implementation of class can be changed at run time which is not possible in inheritance. Encapsulation is maintained in composition but not in inheritance. There will not be any issues between method or the properties while composition is in use.

Hence, composition over inheritance can provide flexibility with extensible and testable code.

# Question 5 b.

It is not ethical for a company to quote a low price for a software contract knowing that the requirements are ambiguous and that they can charge a high price for subsequent changes requested by the customer. When a contract is quoted less, it can be set the bottom price over other company, leading the developer being not well treated than they should deserve.

Now after setting the bottom price the lowest in a contract and charging whole sum to client can make client be aware of that charge. No client will have bad experience and will not refer to that company and probably will share the experience with others too. On the other hand, contractor will also not work for the same company leaving in the middle of the project as they can find better opportunity than that. This will in turn makes company slow on their regular project timeline and can be costly. This can decrease the morale of the employee too and decrease their performance. The company should uphold their normal standards of honesty and integrity and dishonest behaviour of any way that bring disrepute to the software engineering profession.