COSC 735: Software Engineering Test 1 Sample Solution

QUESTION 1:

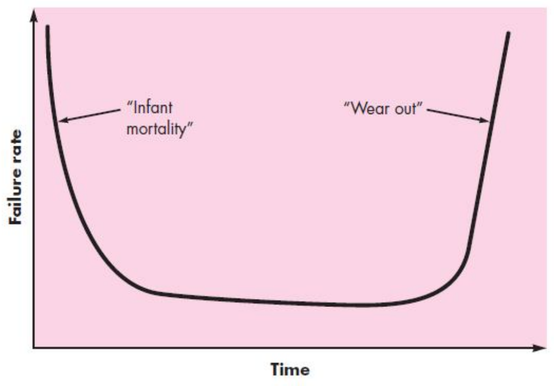
1. **(a) Define Software Engineering.** **[2 marks]**

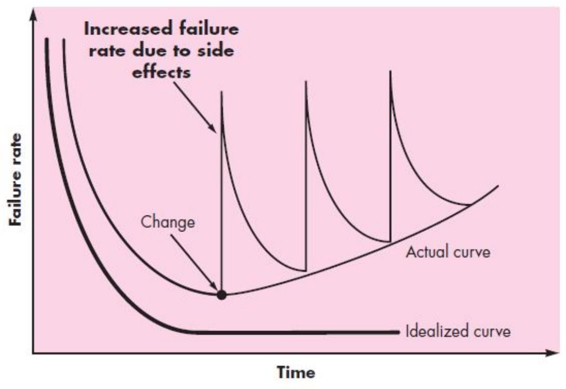
Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

(b) List and explain the characteristics of software that makes it different from hardware. **[6 marks]**

i. Software doesn't wear-out.
The hardware failure rate as a function of time is depicted using the "bathtub curve" in diagram A below. This indicates that hardware exhibit relatively high failure rates early in its life (these failure are often attributable to design or manufacturing defects). Defects are then corrected, and failure rate drops to a steady-state level for some period of time. As time passes, however, the failure rate rises again as hardware components suffer from cumulative effect of dust, vibration, abuse, temperature extremes, and many other environmental maladies.
Stated simple, the hardware begins to wear out.

Software is not susceptible to the environmental maladies that cause hardware to wear out. In theory, therefore, the failure curve for software should take the form of the "idealized curve" as shown in the diagram B below. Undiscovered defects will cause high failure rates early in the life cycle of a program, however, these are corrected and the curve flattens. The idealized curve is a gross over-simplification of actual failure models for software. However, the simplification is clear-software doesn't wear out. But it does deteriorate.





ii. Software is developed or engineered; it is not manufactured in the classical sense.
Although some similarities exist between software development and hardware manufacturing the two activities are fundamentally different. In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are non-existent (or easily corrected) for software. Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different.

iii. Although the industry is moving towards component-based construction, most software continues to be custom built.
In hardware world, component resuse is a natural part of the engineering process. In the software world, it has only begun to be achieved in a broad scale. A software component should be designed and implemented so that it can be reused in many different programs.

(c) Why do requirements change so much? After all, don't people know what they want?
[2 marks]

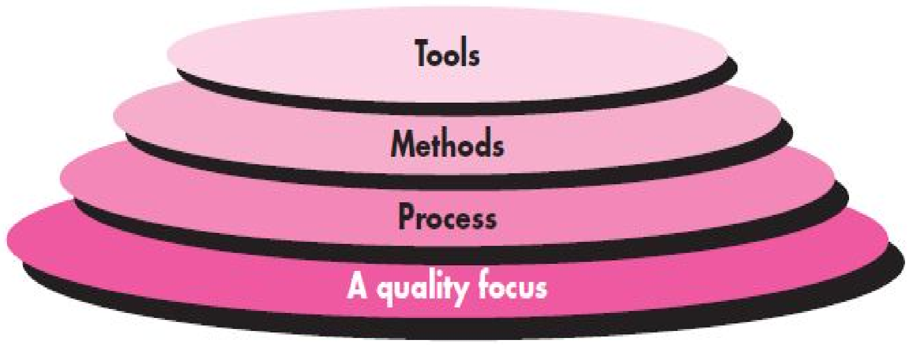
The following are reasons why requirement changes:

i. Missed a requirement. A stakeholder might realize some missing feature of the system
ii. A change can be needed because of a detected defect.
iii. Some mismatch in stakeholder's expectations from what they see in prototypes or working system (when they see it).
iv. The political issues within the organization and between various stakeholders.
v. The marketplace changes.
vi. Law changes. This might change some legal aspects of the system based on what was being made. For example, if a taxation system is needed, Tax policies might change and system might needs to be changed as per the latest laws.

QUESTION 2:

(a) Software engineering is referred to as a layered technology. Explain all the layers with the aid of a diagram. **[2 marks]**

[2 marks]



[4 marks]

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

- Any engineering approach must rest on an organisational commitment to **quality**. The bedrock that supports software engineering is **quality focus**.
- The foundation for software engineering is the "**process layer**". Software engineering process is the glue that holds the technology layers together and enable rational and timely development of computer software. It defines a framework that must be established for effective delivery of software engineering technology.
- Software engineering "**methods**" provides the technical "how to's" for building software. Methods encompass a broad array of tasks that include communication, requirement analysis, design modelling, program construction, testing and support.
- Software engineering "**tools**" provides automated or semi-automated support for the process and the methods.

(b) Extreme Programming (XP) is the most widely used agile process. Explain the key activities / distinguishing features of XP model in Agile. **[4 marks]**

- **XP Planning.** The planning activity (also called *the planning game*) begins with *listening* – a requirements gathering activity that enables the technical members of the XP team to understand the business context for the software and to get a broad feel for required output and major features and functionality.

- **XP Design.** XP design rigorously follows the KIS (keep it simple) principle. A simple design is always preferred over a more complex representation. In addition, the design provides implementation guidance for a story as it is written—nothing less, nothing more. The design of extra functionality (because the developer assumes it will be required later) is discouraged. XP design uses virtually no notation and produces few, if any, work products other than CRC cards and spike solutions, design is viewed as a transient artefact that can and should be continually modified as construction proceeds.

- **XP Coding.** XP Coding Recommends the construction of a unit test for a store *before* coding commences. Another key concept during the coding activity (and one of the most talked about aspects of XP) is *pair programming*. XP recommends that two people work together at one computer workstation to create code for a story. This provides a mechanism for real-time problem solving (two heads are often better than one) and real-time quality assurance (the code is reviewed as it is created).

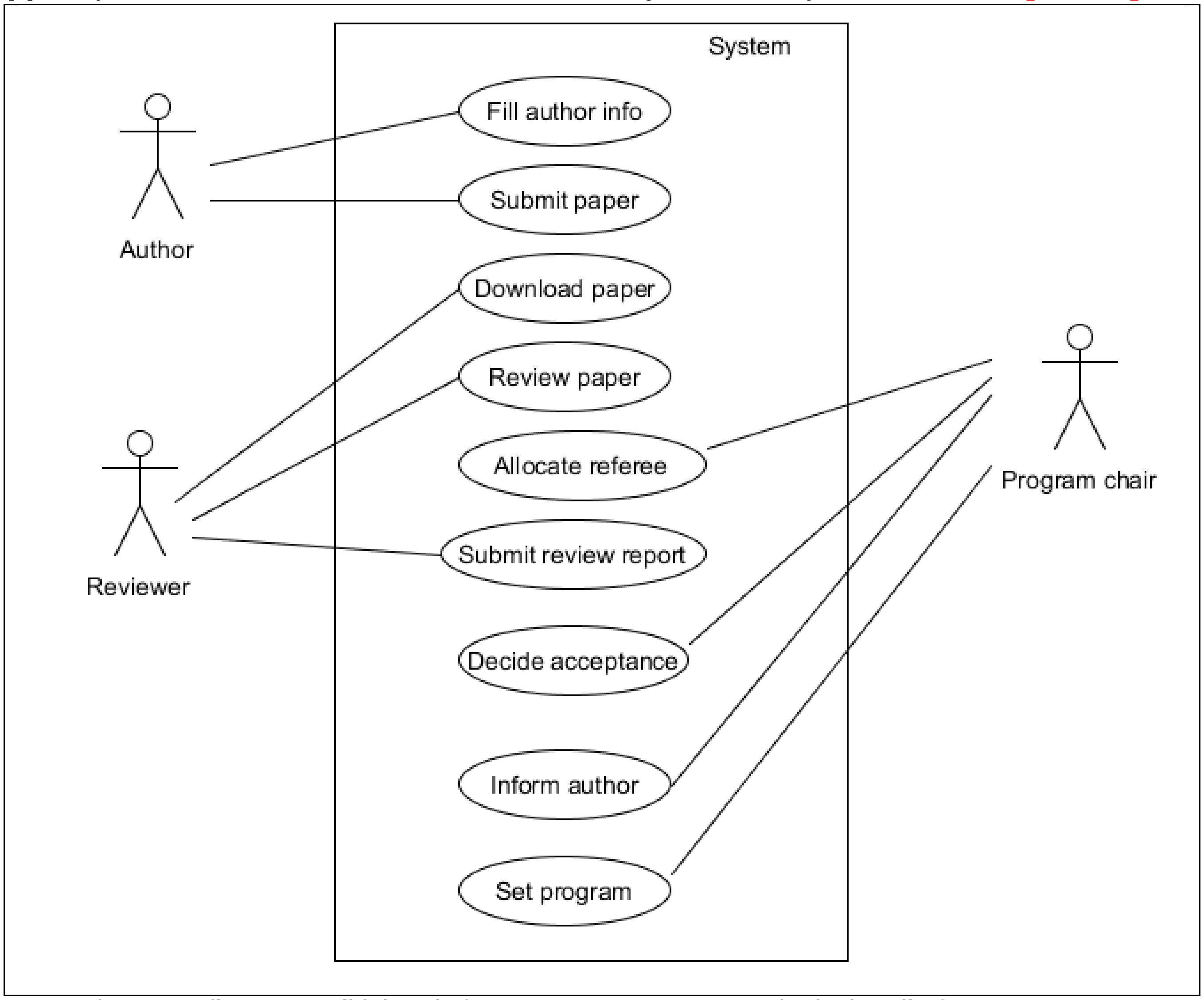
- **XP Testing.** Unit tests are executed daily. The unit tests that are created should be implemented using a framework that enables them to be automated (hence, they can be executed easily and repeatedly). This encourages a regression testing strategy whenever code is modified (which is often, given the XP refactoring philosophy). "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

QUESTION 3:

The following is a narrative description of the business process of organization of conferences with regards to the submitting, reviewing and accepting papers.

The author completes an online form that requests the user to input author name, correspondence address, email and, title of paper. The system validates this data and, if correct, asks the author to submit the paper. The author then browses to find the correct paper on their system and submits it. Once received and stored, the system returns to the author a reference number for the paper. Authors may submit as many papers as they like to be considered for acceptance to the conference up until the deadline date for submissions. Papers are allocated to referees for assessment. They review each paper and submit to the system their decision. Once the programme organiser has agreed the decisions authors are informed by email. Accepted papers are then schedule to be delivered at a conference. This involves allocating a date, time and place for the presentation of the paper.

(a) Analyse the above text and then draw a use case diagram for the system. **[6 marks]**



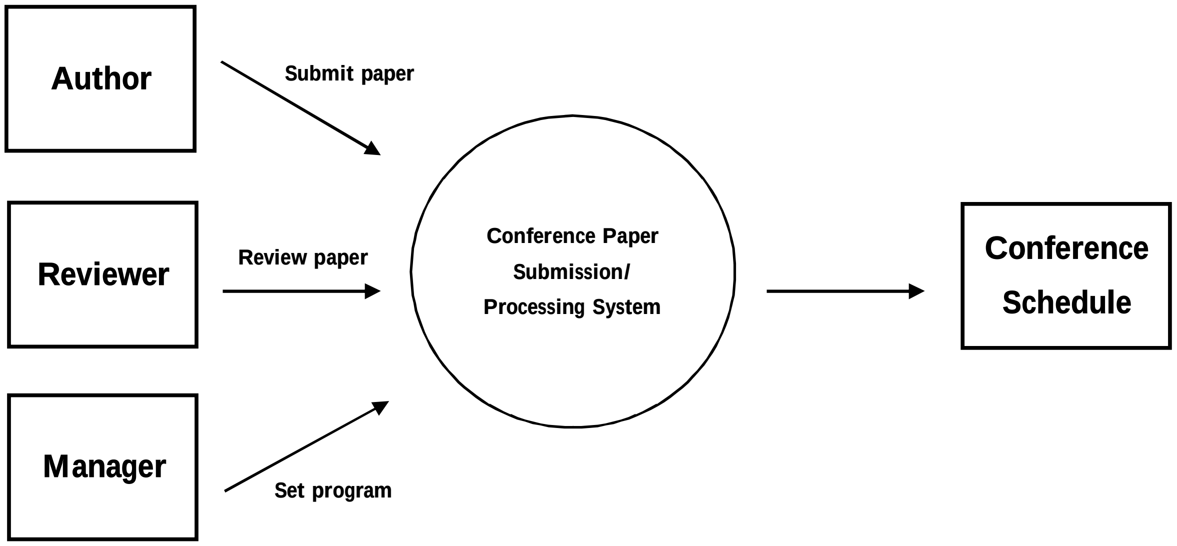
Note: *The use case diagram may slightly varies but, system actors, use cases must be clearly outlined.*

(b) What does Level 0 DFD represent? Using the narrative description above, draw a level 0 DFD for the system. **[4 marks]**

Level 0 DFD: This is the first data flow model (sometimes called a **context diagram**) represents the system as a whole by providing a high level summary of the system input, process and output. Subsequent data flow diagrams refine the context diagram, providing increasing detail with each subsequent level.

[1 marks]

Level 0 DFD for the system: **[3 marks]**



QUESTION 4:

(a) What is Prescriptive Models? **[2 marks]**

Prescriptive process models advocate an orderly approach to software engineering. It stress detailed definition, identification, and application of process activities and tasks. Their intent is to improve system quality, make projects more manageable, make delivery dates and costs more predictable, and guide teams of software engineers as they perform the work required to build a system.

(b) Requirements engineering provides the appropriate mechanism for understanding what the customer wants. Explain any four (4) distinct tasks during requirement engineering. **[8 marks]**

i. Inception	At project inception, you establish a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team.
ii. Elicitation	Elicit requirements from all stakeholders – ask the customer, the users, and others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of the business, and finally, how the system or product is to be used on a day-to-day basis.
iii. Elaboration	The information obtained from the customer during inception and elicitation is expanded and refined during elaboration. This task focuses on developing a refined requirements model that identifies various aspects of software function, behavior, and information.
iv. Negotiation	Agree on a deliverable system that is realistic for developers and customers. You have to reconcile these conflicts through a process of negotiation.
v. Specification	A specification can be a written document, a set of graphical models, a formal mathematical model, collection of usage scenarios, a prototype, or any combination of these.
vi. Validation	The work products produced as a consequence of requirements engineering are assessed for quality during a validation step. Validation is a review mechanism that looks for errors in content, missing information, inconsistencies etc.
vii. Requirements management	Requirements of systems change, and the desire to change requirements persists throughout the life of the system. Requirements management is a set of activities that help the project team identify, control, and track requirements and changes to requirements at any time as the project proceeds.