Study Guide

**Module 1: Introduction**

* The top common 10 questions

1. What is the difference between software engineering and coding assignments? Software engineering is different because the requirements are able to change.
2. What is software?

An executable program.

1. What are good and bad attributes of software?

Does the software meet the user’s needs?

1. What is software engineering?

Design and process of completion.

1. What are the fundamentals software engineering activities?
   1. Software specification
   2. Development
   3. Validation
   4. Software Evolution
2. What is the difference between software engineering and computer science?

Software engineering is the practice. Computer science is the theory.

1. What are the differences between software engineering and system engineer?

Software engineers deal with software and systems engineers deal with hardware.

1. What are the key challenges facing software engineers?

Scalability, accessibility, compatibility, time, security, cost, trust

1. What differences has the internet made to software engineering?

Security/Privacy, social and business requirements, heterogeneous.

1. What are the cost of software engineers?

Specifications and development 60%, validation 40% and Evolution could be double the budget.

1. What are the best software engineering techniques and methods?

Agile, scrum and waterfall.

* Software Engineering Ethics
  + What Ethics means in SWE?

The application of both computer science and engineering philosophy, principles, and practices to the design and development of software systems.

**Module 2: System Models**

* What is it?

A software process that helps create a timely, high quality result.

* Types of Software Process Approaches

Plan driven approach

Agile approach

* Common Software process models, what are they? what are the components? You should be able to compare between them.

**Plan Driven**

**Waterfall Model**

Each phase is one or more documents that should be approved, and the next phase shouldn’t be started until the previous phase has completely been finished.

Customers are consulted at the beginning and more suitable for large systems.

**V-Model (SDLC)**

Considered an extension of the water model. Instead of moving down linearly the process steps allow backtracking.

Customer is only consulted at the beginning and More suitable for smaller systems.

**Agile**

**Incremental Model**

Develops an initial implementation exposing this to user feedback and evolving it through several versions until an acceptable system has been developed.

Flexible allowing customer to see product development with early rollouts.

**Spiral Model**

Risk driven software development process model. Based on the unique risk patterns of a given project the spiral model guides a team to adopt elements of one or more process models such as incremental, waterfall or evolutionary prototyping.

Great for high risk projects and is suitable for large projects.

**Integration and Configuration Model (Reusable Model)**

Used where systems are integrated from existing components.

* The basic process activities of any Software Module( what are they? should be able to identify, compare and define all the subcomponents)

Specification -> Design -> Validation -> evolution.

**Module 3: Agile**

(Review the Online modules and the ScrumReferenceCard.pdf)

* What is Scrum?

Management framework for incremental product development using one or more teams of seven-ish people. Provides a structure of roles, meetings, rules, and artifacts. Teams are responsible for creating and adapting their processes. Uses sprints that are no more than 30 days long. Teams try to develop a releasable product increment every Sprint.

* What are the main Artifacts, Ceremonies in the scrum?

**Artifacts** –

* **Product backlog** is the set of all baseline requirements.
* **Sprint backlog** is the subset of the product backlog.
* **Product increment** is when a project is potentially shippable and aligns with the “definition of done”.

**Ceremonies** – Answered 2 questions below.

* Main People Role
* **Product Owner** is the sole person responsible for managing the product backlog.
* **Scrum Master** is a servant-leader for the scrum team and helps those outside the scrum team understand which of their interactions with the scrum team are helpful and which aren’t.
* **The Development Team** does the work of delivering a potentially releasable increment of “done” product at the end of each sprint.
* You should be able to identify the ceremonies details and any other questions related to them.
* **Sprint planning** describes and presents the ordered product backlog and sprint goal. Time: 4 hours x2.
* **Daily scrum meeting** identifies & removes impediments to development. Time: 15 minutes.
* **Sprint review** allows the team to demonstrate the increment with focus on the sprint goal. Time: 1.5 hours x2.
* **Sprint retrospection** revises the way of work in the past in order to make it more efficient and effective in the future. Time: 2 hours x2.

**Module 4: Requirements**

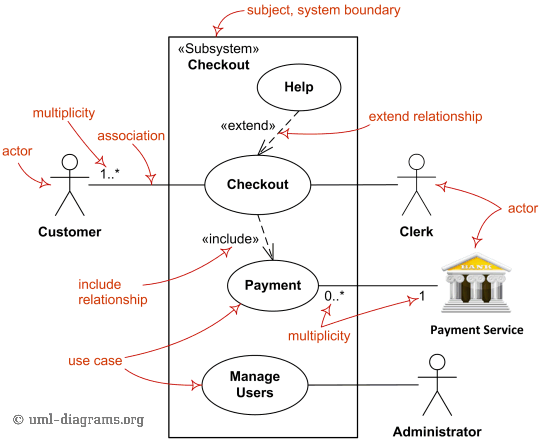
* What is functional Requirements, Non-Functional Requirements?
* **Functional requirements** are those which are related to the technical functionality of the system.
* **Non-functional requirements** are ones that specifies criteria that can be used to judge the operation of a system in particular conditions, rather than specific behaviors.
* What are Good requirements and what is Bad ones?
* **Good requirements:** Unambiguous, Testable, Clear.
* What is user stories, What is a use case?
* **User story** is a tool used in agile software development to capture a description of a software feature from an end-user perspective. Describes the type of user, they want and why. Helps create a simplified description of a requirement.
* **Use Case** is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal. The actor can be a human or other external system.
* How to write user stories

As---- A user

I need---- to be able to remove items from my cart

Because---- I no longer want or need them.

* Use case Model diagram (interpret and draw).



* Use Case Narrative details

Explains how the use case, in essence, describes the interaction between an actor to achieve a goal of observable value.

**Module 5: Unit Test and Junit**

* What are the different System test phases?
* Unit Test / System Test
* Release Test
* User Test – Feedback, Alpha, Beta
* How to write a Unit test case using Junit Framework and Hamcrest?

@Test  
@DisplayName("Times 10000 objects being pushed into linked stack.")  
public void push10000Test()  
{  
 LinkedStackFixed linkedStackFixed = new LinkedStackFixed();  
 double startTime = System.*currentTimeMillis*();  
 for(int i = 0; i < 10000; i++)  
 {  
 linkedStackFixed.push(i);  
 }  
 double endTime = System.*currentTimeMillis*();  
 double totalTime = endTime - startTime;  
 System.*out*.println("Total Time: " + totalTime + " Milliseconds");  
 *assertTimeout*(Duration.*ofMillis*(5), () -> {  
 for(int i = 0; i < 10000; i++)  
 {  
 linkedStackFixed.push(i);  
 }});  
}

* How to work TDD?
* An iterative development process. Each iteration starts with a set of tests written for a new piece of functionality.
* Benefits of TDD
  1. Unit test proves that the code actually works.
  2. Can drive the design of the program.
  3. Refactoring allow to improve the design of the code.
  4. Low level regression test suite.
  5. Test first reduce the cost of the bugs.
* Drawbacks of TDD
  1. Developer can consider it as a waste of time.
  2. The test can be targeted on verification of classes and methods and not on what the code really should do.
  3. Test become part of the maintenance overhead of a project.
  4. Rewrite the test when requirements change.
* Phase 1 (Requirement Definition)
* Phase 2 (Executing Tests)
* Phase 3 Adding/Refactoring Code

**Module 6: System Architecture Design**

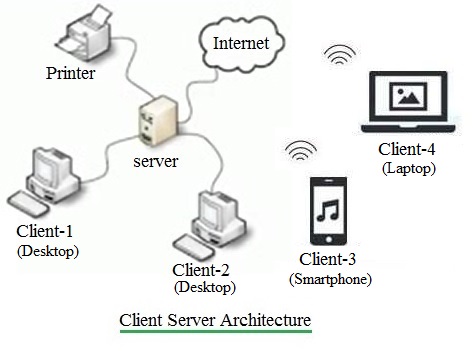
* What are the common traditional Architecture pattern designs?
* **Model View Controller (**MVC) is commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the way’s information is presented to and accepted from the user. This design pattern decouples these major components allowing for efficient code reuse and parallel development.



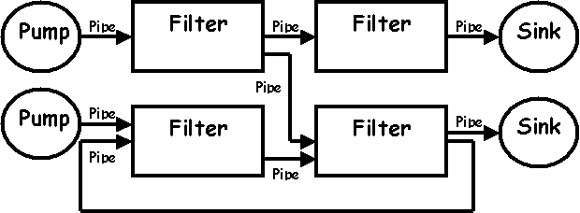
* **Layered Architecture** pattern is the separation of concerns among components. Components within a specific layer deal only with logic that pertains to that layer.



* **Client Server Architecture** is a model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server of a network or internet connection.



* **Pipe and Filter Architecture** has any number of components (filters) that filter data, before passing it on via connectors (pipes) to other components. The filters are all working at the same time. The architecture is often used as a simple sequence, but it may also be used for very complex structures.



The output of one program can be the input of another.

* You should be able to think about Architecture design for a given system using one or a combination of the known Architecture Patterns and show all the steps involved to come up with the design.
* Architecture patterns with Cloud Services and definitions for the new term.