

**13/06/2019**

**UNITED STATES INTERNATIONAL UNIVERSITY (USIU) –AFRICA**

**APT 2080**

**Mrs. ESTHER GACHANGA**

**NAME: IAN KATHURIMA MWITI**

**ID NUMBER: 658291**

**US2019**

**COURSE SUMMARY (Week 1- week 6)**

On week one, the class commenced with a self-introduction followed by the lecturer introducing herself and beginning on the topic **Introduction to Software Engineering**. We tackled the definition of a Software as instructions that when executed provide a desired function, the applications, multiple types of computing such as ubiquitous and open world. We defined what a legacy system as a software that has been around a relatively long period and still serves its purpose or business need, and its importance to the environment, mentioning M-Pesa as a valid example to currently relate to. The definition of a software engineering was mentioned which was the application of a systematic, disciplined approach to the development, operation and maintenance of software. The process model adaptation is the degree to which work products are identified and required.

On week two, we discussed the **Software Process** as the structured set of activities required to develop a software system. The software process model is an abstract representation of a process. There are two types of processes which are plan-driven (activities planned in advance) and agile (incremental planning and changes are easier to make). There are 3 types of process models, the waterfall model, incremental development model and reuse oriented software engineering model. Software specification is defined as the process of establishing what services are required on the system’s operation and development. Software verification and validation are important because they show that the system conforms to its specification and meets the user’s requirements. Software evolution is the changing of an existing system requirement(s) to meet new requirement(s).

On week three, we got into **Requirements Engineering**, where we defined software requirements document as the official statement of what is required of the system developers. Its users are people such as system users/customers, managers, system engineers, test engineers and maintenance engineers. The document structure was well broken down, paying attention to the system requirements specification, system models and appendices sections. Requirements specification is the process of writing the user & system requirements in a document, with multiple guidelines to aid the writer such as using consistent language and avoiding computer jargon. Requirements engineering has 4 processes namely feasibility study, requirements elicitation, requirements specification and requirements validation. The elicitation and analysis has stages, like requirements discovery involving identifying requirements, requirements classification and organization, requirements prioritization and negotiation and requirements specification. Requirements checking has multiple steps such as validity, consistency, completeness, realism and verifiability checks. Validation techniques include requirement reviews, prototyping and test case generation. Review checks include verifiability, traceability, comprehensibility and adaptability checks. We also discussed some of the system documentation tools such as narratives, flowcharts, diagrams and other written materials.

On week four, we tackled **Software Architectural Design,** stating the architecture is important because it represents the software’s architecture easily and highlights the early design decisions. The various architectural styles are data centered, data flow, cell and return, object oriented and layered architectures. Each system architecture encompasses a set of components, a set of connectors, connectors and semantic models. The architectural description language (ADL) provides a semantic and syntax for describing a software architecture. It provides the designer with the ability to decompose architectural components, represent interfaces and compose individual components into larger architectural blocks. Factoring is the process of determining what properties and methods belong on an interface, and refactoring as the process of changing a software system in such a way that it doesn’t alter its external behavior. We then began on **System Modelling,** beginning on Unified Modelling Language (UML) as the standard language for specifying, visualizing, constructing and documenting artifacts of a software system. The goal of modelling software is to explain complicated lines of code into and understandable graphical representation. Using a fitting example, we used an aircraft as a model (abstraction describing subset of system0, the flight simulator as a view (depicts selected aspects of model), and blueprints as a notation (graphical or textual rules for depicting views). The major elements of UML are building blocks (things, relationships and diagrams), common mechanics of UML and the rules to connect building blocks. DFD is a graphical modeling tool that depicts the flow of data.

On week 5, we started on **Software Testing Part One,** defining testing as the act of identifying whether a program once executed it performs what is intended to do, also discover faults in the system, and demonstrate to the customers and developer that the software meets its requirements. Verification is done to discover faults and validation is done to make sure the software meets its requirements; the general main aim being is to establish confidence that the system is “fit for purpose”.