TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES VISAYAS
Capt. Sabi St., City of Talisay, Negros Occidental

OFFICE OF THE COLLEGE DEAN

F-PCO -12
2
23 Jun 2022

College of Engineering Technology
Computer Engineering Technology Department
(End-User)

PRELIM EXAM
(Term)

COMP 332B –SOFTWARE ENGINEERING
(Subject Code and Descriptive Title)

3ND TERM, 2022-2023
(Term) (SY)

/55

Name: CLARION REYSIL M. Yr. & Sec. TO9-A Score
Last Name First Name M.I.

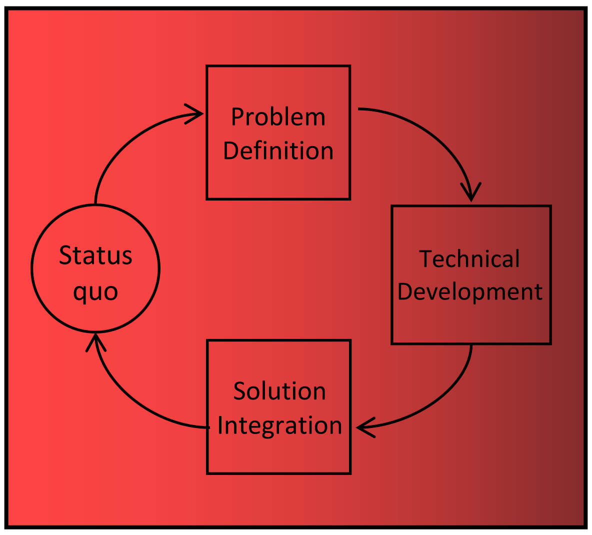
Instructor: RODELEYN_TOLENTINO Proctor: ______ Date: 05/23/2023

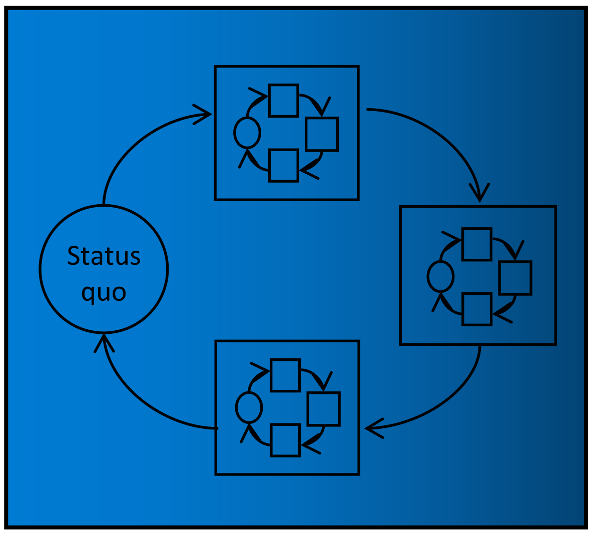
Instructions: Read the instructions carefully. No Erasures.

Test I. Discuss and Draw the following. ***[5 points each]***

1. Software process model

- A software process model, also known as a software development life cycle (SDLC), is a structured approach that defines the various stages and activities involved in the development and maintenance of software systems. These models provide a framework for managing and organizing the software development process, ensuring that it progresses smoothly and meets the desired objectives. There are several different software process models, and the choice of model depends on factors such as project requirements, team size, time constraints, and customer collaboration preferences. In this discussion, I will cover some commonly used software process models.

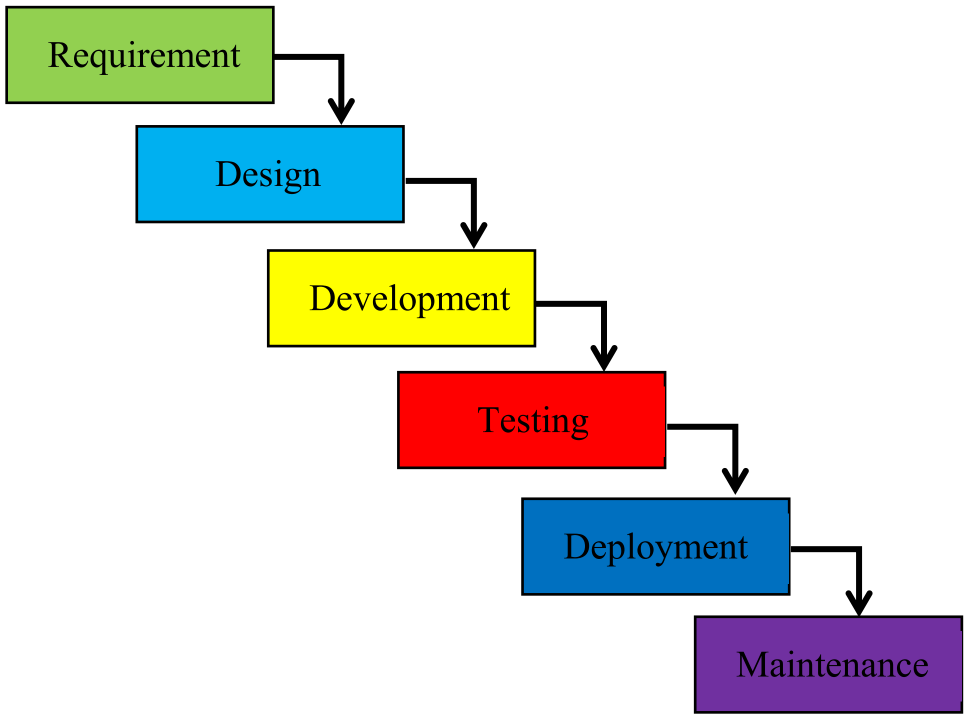




SOFTWARE PROCESS MODEL

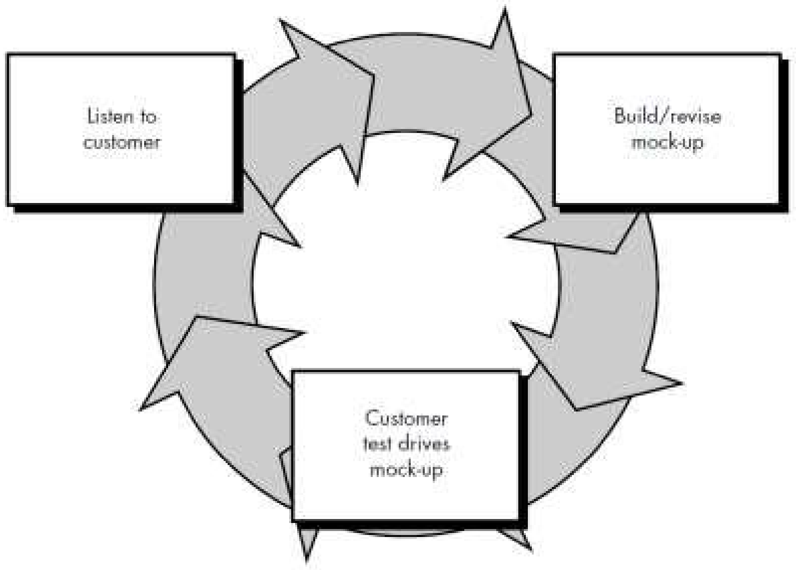
2. Linear sequential model

- The linear sequential model, also known as the waterfall model, is a software development process that follows a strict linear and sequential approach. It consists of distinct phases, such as requirements gathering, design, implementation, testing, deployment, and maintenance. Each phase is completed before moving on to the next, and there is little room for iteration or changes once a phase is completed. The linear sequential model is suitable for projects with well-defined and stable requirements, where changes are unlikely to occur. However, it may not be suitable for projects that require flexibility or extensive customer collaboration.



THE LINEAR SEQUENTIAL MODEL

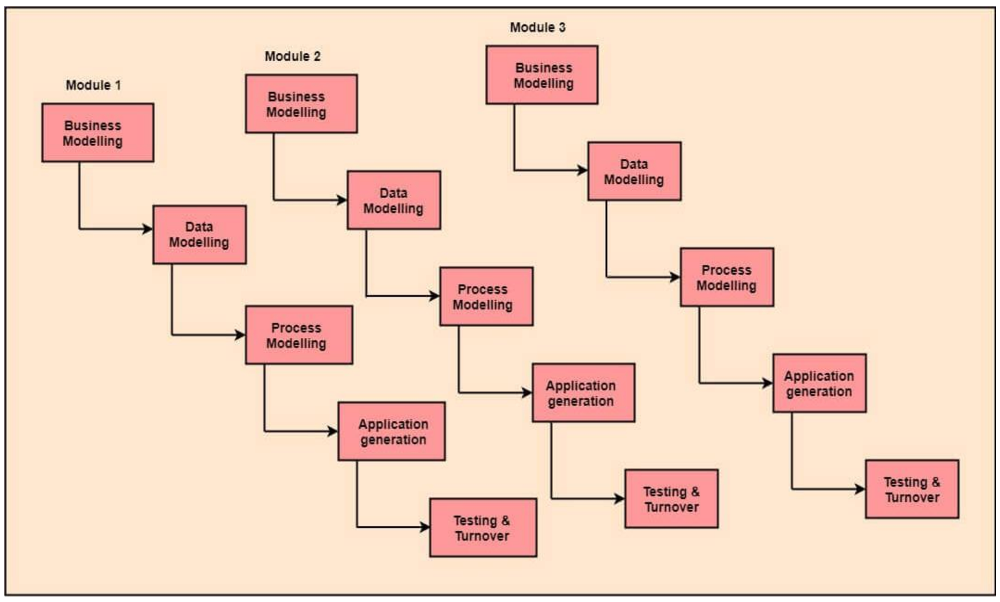
3. Prototyping Model
The prototyping model is an iterative software development process that focuses on quickly creating and refining prototypes of the software system. It involves developing a basic working model or prototype that demonstrates key functionalities and features of the final product. This prototype is then evaluated and refined based on user feedback and requirements. The process continues with multiple iterations of prototyping, incorporating feedback and making improvements until the final system is developed. The prototyping model is beneficial when requirements are not well-defined or when user involvement and feedback are crucial in shaping the final product.



THE PROTOTYPING MODEL

4. RAD Model

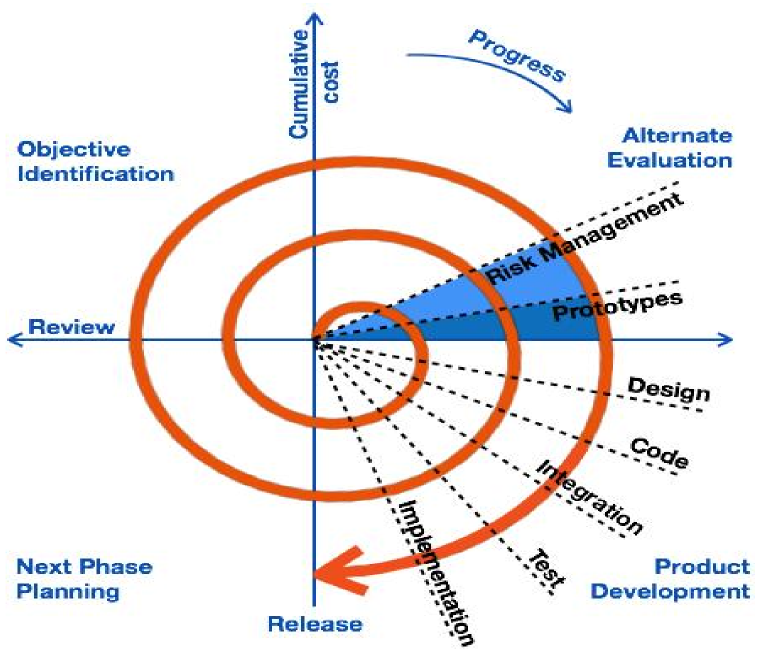
The RAD (Rapid Application Development) model is an iterative and incremental software development process that emphasizes quick development and delivery of working software. It focuses on active user involvement, prototyping, and iterative feedback cycles. The RAD model typically consists of four phases: requirements planning, user design, construction, and cutover. It is particularly useful for projects with well-defined business objectives, tight timeframes, and where early and frequent user feedback is essential. The RAD model promotes faster development and allows for flexibility and adaptability in responding to changing requirements.



THE RAD MODEL

5. Spiral Model

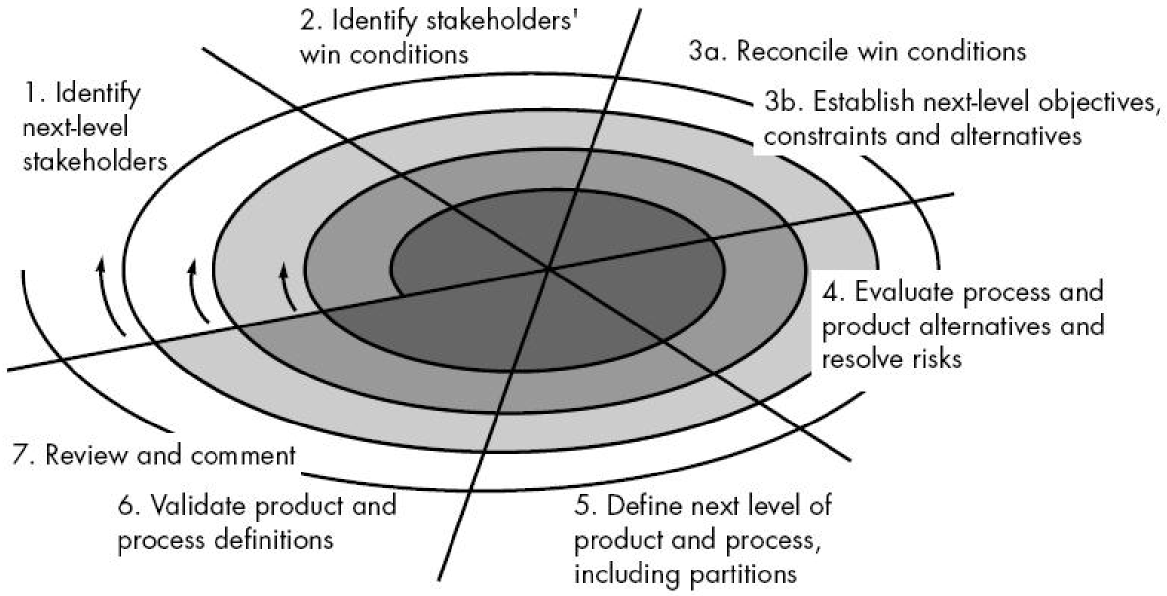
- The Spiral Model is a risk-driven software development process model that combines elements of both the waterfall model and iterative development. It involves a series of iterative cycles, each representing a spiral, which starts with initial requirements and progresses through planning, risk analysis, engineering, and evaluation. The primary feature of the Spiral Model is its focus on risk management, as each spiral iteration includes risk identification, assessment, and mitigation strategies. This model allows for incremental development and emphasizes early prototyping and customer involvement. The Spiral Model is well-suited for large and complex projects where risks and uncertainties need to be effectively managed throughout the development life-cycle.



THE SPIRAL MODEL

6. Winwin Spiral Model

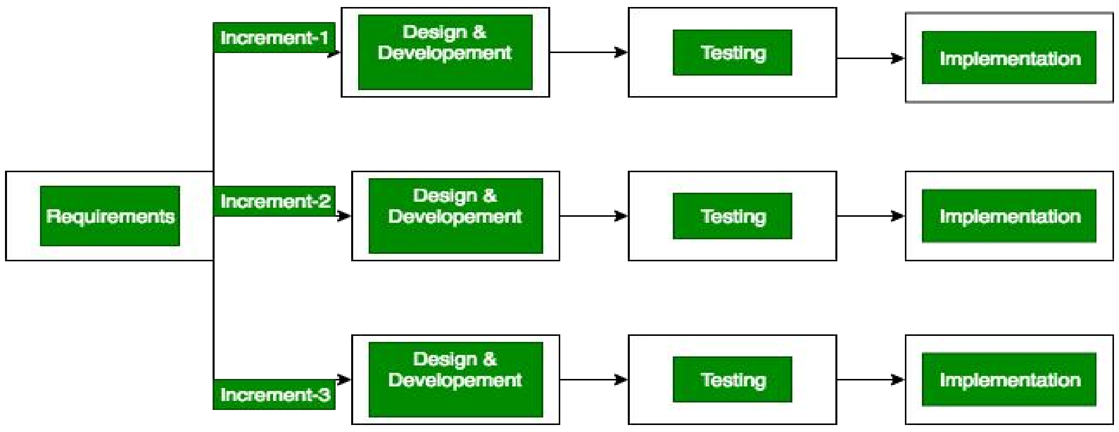
- The Winwin Spiral Model is an extension of the traditional Spiral Model that incorporates the concept of "win-win" negotiations and collaboration between stakeholders. It aims to address the challenges of balancing conflicting interests and requirements among different stakeholders in a software development project. The Winwin Spiral Model includes additional activities focused on communication, negotiation, and consensus building. These activities are designed to foster collaboration, resolve conflicts, and ensure that the needs of all stakeholders are considered and satisfied. By emphasizing open communication and shared decision-making, the Winwin Spiral Model promotes a collaborative and cooperative approach to software development, leading to improved stakeholder satisfaction and project success.


2. Identify stakeholders' win conditions
3a. Reconcile win conditions
1. Identify next-level stakeholders
3b. Establish next-level objectives, constraints and alternatives
4. Evaluate process and product alternatives and resolve risks
7. Review and comment
6. Validate product and process definitions
5. Define next level of product and process, including partitions

THE WINWIN SPIRAL MODEL

7. Incremental Model

- The Incremental Model is an iterative software development process model where the software system is developed and delivered in increments or smaller modules. It involves breaking down the project into multiple manageable portions, with each increment delivering a functional and executable piece of software. The development process progresses incrementally, adding new features and functionality with each iteration. This model allows for early delivery of working software and facilitates frequent feedback from stakeholders. It also provides the flexibility to incorporate changes and improvements based on the feedback received, resulting in a more refined and tailored final product. The Incremental Model is particularly useful when requirements are subject to change, and it allows for better risk management and early validation of critical features.



THE INCREMENTAL MODEL

Test II. Discuss the Attributes of good software ***[10 points]***

Good software possesses several key attributes that contribute to its overall quality and effectiveness. Firstly, one important attribute is functionality. Good software should fulfill its intended purpose and meet the needs and expectations of its users. It should provide the necessary features and capabilities to perform the desired tasks efficiently and effectively. Secondly, usability is another vital attribute of good software. It should be user-friendly and intuitive, allowing users to interact with it easily and efficiently. Clear and well-designed interfaces, logical workflows, and appropriate feedback mechanisms are essential to enhance usability. Reliability is another crucial attribute of good software. It should be stable and dependable, consistently delivering the expected results without errors or failures. Reliability can be achieved through rigorous testing and quality assurance processes, as well as robust error handling and fault tolerance mechanisms.

Good software should also exhibit high performance. It should be efficient and responsive, delivering quick results and utilizing system resources optimally. Performance optimizations and efficient algorithms can contribute to achieving high-performance levels. Maintainability is another attribute of good software. It should be modular and well-structured, allowing for easy maintenance, updates, and enhancements. Clean and readable code, along with documentation and version control, facilitate maintainability and support future modifications. Lastly, good software should prioritize security. It should protect sensitive data, prevent unauthorized access, and withstand potential security threats. Implementing secure coding practices, encryption mechanisms, and regular security audits are crucial to ensure the software's security.