****

**GARY MAGADZIRE SCHOOL OF AGRICULTURE AND NATURAL SCIENCE**

**Bachelor of Science Honors in Computer Science**

**(BLOCK RELEASE)**

**PROGRAM: COMPUTER SCIENCE**

**LEVEL: 1.2**

**MODULE: HCS 125**

**ASSIGNMENT: 01**

**LECTURER: Ms Madyembwa**

**Group assignment for:**

**Prince chikwature m231063**

**Andrew machokoto m226116**

**Software Process Models and Milestone**

Software processes are the activities for designing, implementing, and testing a software system. The software development process is complicated and involves a lot more than technical knowledge. That’s where software process models come in handy. A software process model is an abstract representation of the development process. Software process models are systematic approaches used to guide the development of software. They provide a framework for managing the various activities and tasks involved in software development. Milestones, on the other hand, are specific points in the software development process that represent significant achievements or stages of completion. They serve as checkpoints to assess progress and ensure that the project is on track.

**THE FOLLOWING ARTICLE WILL PROVIDE AN OUTLINE ON THE QUESTIONS BELOW**

* [What is a software process model?](https://www.educative.io/blog/software-process-model-types#what)
* What are milestones?
* [Factors in choosing a software process](https://www.educative.io/blog/software-process-model-types#factors).
* [Types of software process models](https://www.educative.io/blog/software-process-model-types#types) and their milestones.

**What is a software process model?**

A software process model is an abstraction of the software development process. The models specify the stages and order of a process.

**A model will define the following:**

* The tasks to be performed
* The input and output of each task
* The pre and post-conditions for each task
* The flow and sequence of each task

The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

**SOFTWARE PROCESS MILESTONE**

* Software process milestones are significant events or achievements that mark the completion of specific phases or tasks within the software development life cycle **(SDLC).**
* Importance of Milestones:  
   Provide clear progress indicators for project tracking.

Enable effective communication between stakeholders.  
Facilitate risk assessment and mitigation.  
Ensure timely delivery of project deliverables.  
Aid in resource allocation and planning.

* Common Milestones:  
  a) Requirements Gathering:  
  - Elicitation of user needs and system requirements.  
  - Documentation of functional and non-functional requirements.  
    
  b) Design Phase:  
  - Creation of system architecture and detailed design specifications.  
  - Development of prototypes or wireframes for user interface design.  
    
  c) Development and Coding:  
  - Implementation of the software based on the design specifications.  
  - Adherence to coding standards and best practices.  
    
  d) Testing and Quality Assurance:  
  - Verification and validation of the software against requirements.  
  - Identification and resolution of defects or bugs.  
    
  e) Deployment and Release:  
  - Preparation of the software for production environment.  
  - Installation, configuration, and user training.  
    
  f) Maintenance and Support:  
  - Post-deployment activities, including bug fixes, updates, and enhancements.  
  - Monitoring and addressing user feedback.

**FACTORS TO CONSIDER IN CHOOSING SOFTWARE PROCESS**

**PROJECT REQUIREMENTS:** Before you choose a model, take some time to go through the project requirements and clarify them alongside your organizations or team’s expectations. Will the user need to specify requirements in detail after each iterative session? Will the requirements *change* during the development process?

**PROJECT SIZE:** Consider the size of the project you will be working on. Larger projects mean bigger teams, so you’ll need more extensive and elaborate project management plans.

**PROJECT COMPLEXITY:** Complex projects may not have clear requirements. The requirements may change often, and the cost of delay is high. Ask yourself if the project requires constant monitoring or feedback from the client.

**COST OF DELAY:** Is the project highly time-bound with a huge cost of delay, or are the timelines flexible?

**CUSTOMER INVOLVEMENT:** Do you need to consult the customers during the process? Does the user need to participate in all phases?

**FAMILIARITY WITH TECHNOLOGY:** This involves the developers’ knowledge and experience with the project domain, software tools, language, and methods needed for development.

**PROJECT RESOURCES:** This involves the amount and availability of funds, staff, and other resources.

**Types of software process models**

As we mentioned before, there are multiple kinds of software process models that each meet different requirements. Below, we will look at the top seven types of software process models that you should know.

**Waterfall Model**

The Waterfall model follows a sequential approach with distinct phases that must be completed before moving to the next one. It is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.

**Milestones For the Waterfall Model**

* Requirements
* Design
* Implementation
* Testing
* Deployment
* Maintenance

**Requirement gathering**: Gathering and documenting the software requirements from stakeholders.

**System design**: Creating a high-level design that outlines the system architecture, components, and their interactions.

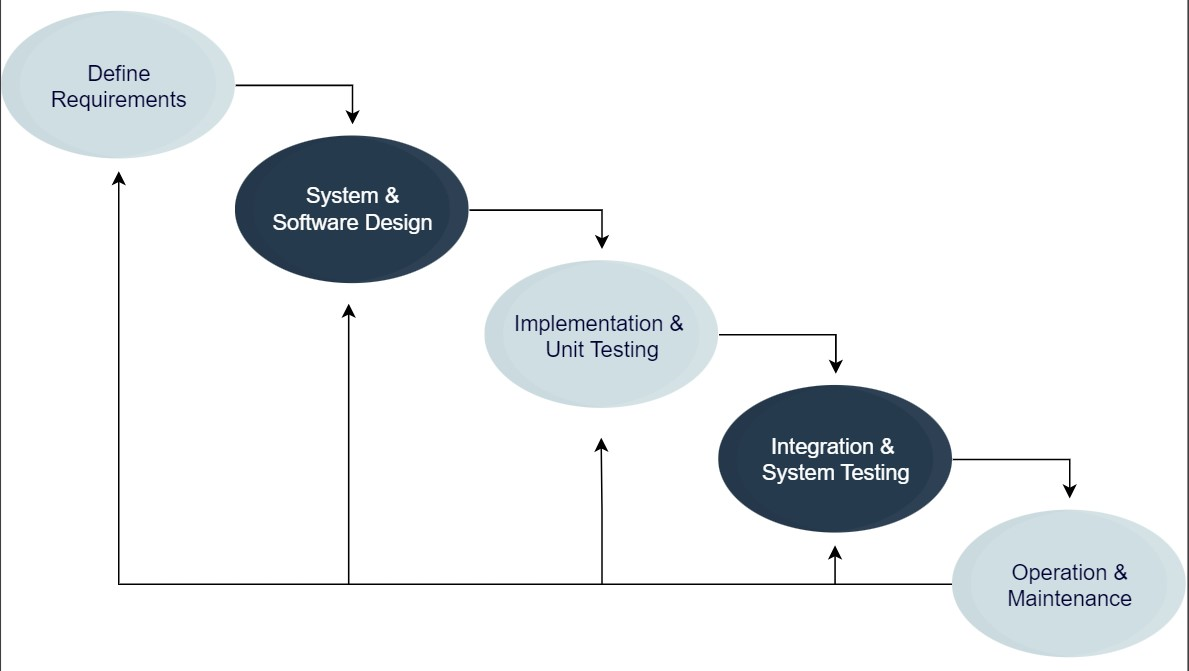
**Detailed design**: Developing detailed designs for each component or module, specifying how they will be built.

**Implementation**: Writing code and building the software system based on the detailed designs.

**Testing**: Conducting various levels of testing, including unit testing (testing individual components), integration testing (testing the interaction between components), and system testing (testing the complete system).

**Deployment**: Releasing the software to the users or customers.

**Maintenance**: Providing ongoing support, bug fixes, and updates based on user feedback and discovered issues.



**The waterfall model**

Each of these phases produces one or more documents that need to be approved before the next phase begins. However, in practice, these phases are very likely to overlap and may feed information to one another. The software process **isn’t linear**, so the documents produced may need to be modified to reflect changes. The waterfall model is easy to understand and follow. It doesn’t require a lot of customer involvement after the specification is done. Since it’s inflexible, it can’t adapt to changes. There is no way to see or try the software until the last phase. The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to radically change.

**V Model**

The V-Model emphasizes the relationship between each development phase and its corresponding testing phase**.** The V model (Verification and Validation model) is an extension of the waterfall model. All the requirements are gathered at the start and cannot be changed. You have a corresponding testing activity for each stage. For every phase in the development cycle, there is an associated testing phase.

**Milestones The V Model**

Requirements specification: Gathering and documenting the software requirements.

System design: Creating a detailed design that describes the system architecture and components.

Architecture design: Designing the software architecture based on the system design.

Module design: Designing individual modules or components based on the architecture design.

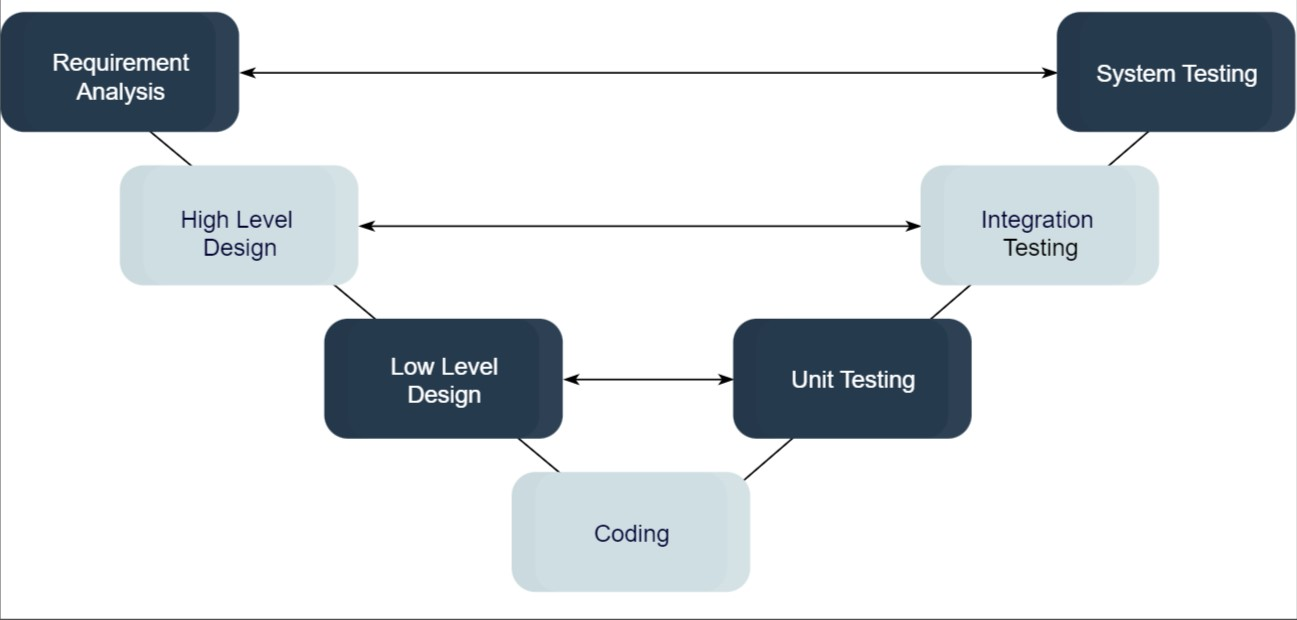
Coding/Implementation: Writing code and building the software modules.

Unit testing: Testing individual modules to ensure they function correctly.

Integration testing: Testing the integration and interaction between modules.

System testing: Testing the complete system to verify that it meets the specified requirements.

Acceptance testing: Testing the system with user involvement to ensure it meets user expectations



**The V Model**

The corresponding testing phase of the development phase is planned in parallel, as you can see above. The V model is highly disciplined, easy to understand, and makes project management easier. But it isn’t good for complex projects or projects that have unclear or changing requirements. This makes the V model a good choice for software where downtimes and failures are unacceptable.

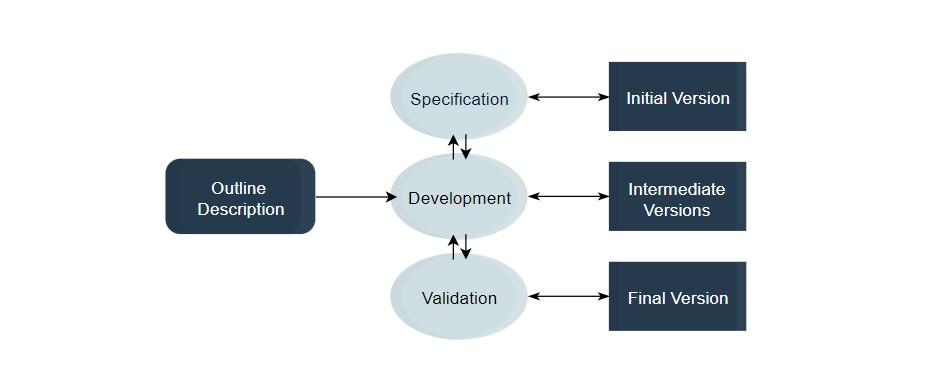
**Incremental Model**

The Incremental model involves dividing the software development process into smaller increments or modules that are delivered one after the other in quick succession. The most important functionality is implemented in the initial increments. The subsequent increments expand on the previous ones until everything has been updated and implemented.

Incremental development is based on developing an initial implementation, exposing it to user feedback, and evolving it through new versions. The process’ activities are interwoven by feedback.

Each iteration passes through the requirements, design, coding, and testing stages.

Each iteration passes through the requirements, design, coding, and testing stages.



The incremental model lets customers and developers see results with the first increment. If the customers don’t like anything, everyone finds out a lot sooner. The incremental model is efficient because the developers only focus on what is important and bugs are fixed as they arise, but you need a clear and complete definition of the whole system before you start.

The incremental model is great for projects that have loosely coupled parts and projects with complete and clear requirements.

**Incremental Model Milestones**

Requirements analysis: Identifying and documenting the user requirements for the initial increment.

Design and development: Designing and developing the first increment, which includes specific features or functionality.

Testing and integration: Testing the developed increment and integrating it with previously developed increments.

Deployment: Deploying the completed increment to users or customers.

Ongoing iterations: Repeating the cycle for subsequent increments, adding new features or functionality based on user feedback and requirements.

Feature-Driven Development (FDD):

The Feature-Driven Development (FDD) model focuses on feature delivery and collaboration among team members:

Develop overall model: Creating an overall model that represents the system's structure and features.

Build feature list: Identifying and prioritizing the features to be implemented.

Plan by feature: Planning the development tasks and schedules for each feature.

Design by feature: Designing and developing features incrementally, focusing on one feature at a time.

Build by feature: Implementing and testing each feature individually.

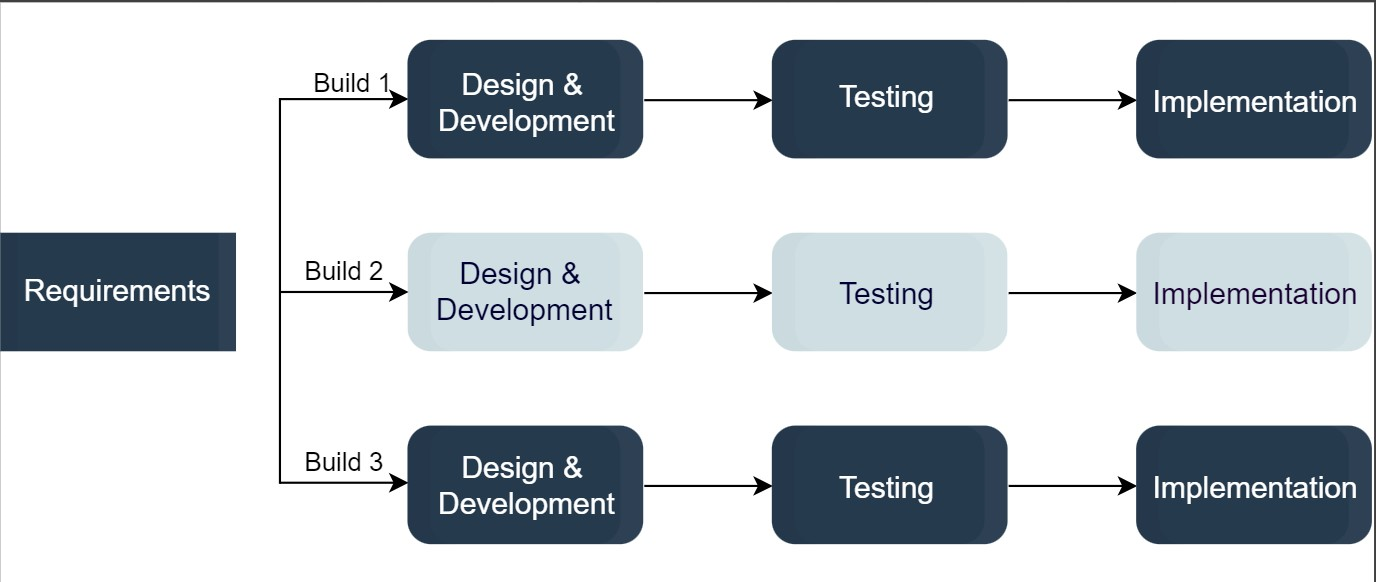
Test by feature: Conducting feature-level testing to ensure their functionality and integration.

**Iterative Model**

Iterative models involve repetitive cycles of development, where each cycle produces a working and tested software increment.The iterative development model develops a system by building small portions of all the features. This assist to meet the initial scope quickly and release it for feedback. In the iterative model, you start off by implementing a small set of software requirements. These are then enhanced repeated in the evolving versions until the system is completed. This process model starts with part of the software, which is then implemented and reviewed to identify further requirements.

Like the incremental model, the iterative model allows you to see the results at the early stages of development. This makes it easy to identify and fix any functional or design flaws. It also makes it easier to manage risk and change requirements.

The deadline and budget may change throughout the development process, especially for large complex projects. The iterative model is a good choice for large software that can be easily broken down into modules.



**Milestones for the iterative Model**

Inception: Defining the project scope, objectives, and high-level requirements to establish the project's feasibility.

Elaboration: Analysing the requirements in detail, creating a more refined design, and developing a prototype or initial version of the software.

Construction: Developing and testing the software incrementally in iterations, adding new features and refining existing ones.

Transition: Preparing the software for deployment, including final testing, user training, and documentation.

Release: Deploying the software to the users, typically in a controlled manner.

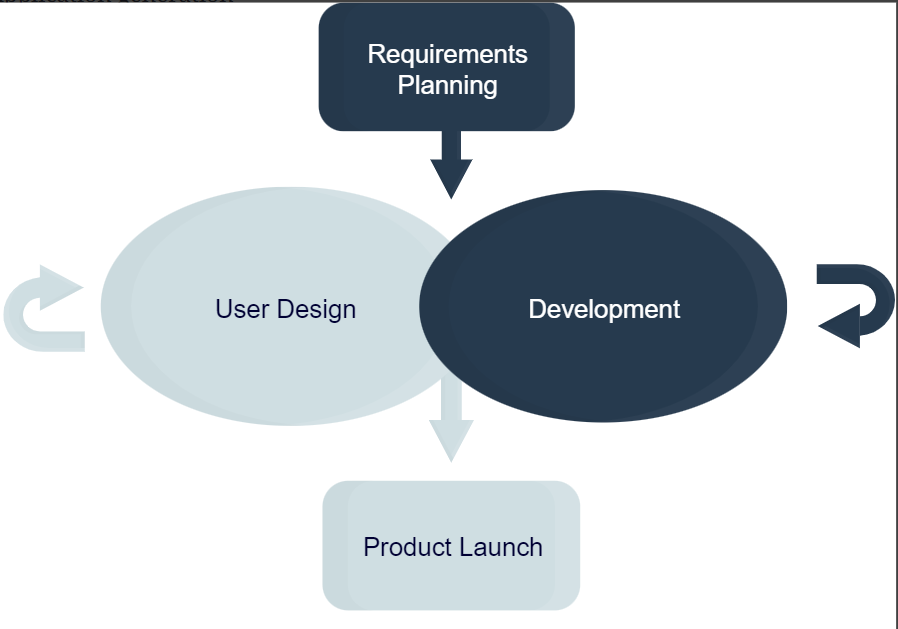
Ongoing iterations: Incorporating user feedback, making further enhancements, and repeating the iterative cycle.

**RAD Model**

The RAD model focuses on rapid prototyping and iterative development to quickly deliver software solution**.** It is based on iterative development and prototyping with small planning involved. You develop functional modules in parallel for faster product delivery. It involves the following phases:

* Business modelling
* Data modelling
* Process modelling
* Application generation
* Testing and turnover

RAD concept focuses on gathering requirements using focus groups and workshops, reusing software components, and informal communication.



The RAD model allows changing requirements, reduces development time, and increases the reusability of components. The rapid application development can be complex to manage. Hence, the RAD model is ideal for systems that need to be produced in a short time and have known requirements.

**Milestones for the Rapid Application Development**

**Requirements planning**: Identifying and prioritizing user requirements.

**User design**: Involving users in the design process to gather feedback and create prototypes.

**Rapid construction**: Developing the software incrementally, integrating user feedback and creating prototypes in short iterations.

**Cutover**: Deploying the software incrementally, ensuring a smooth transition from the old system to the new one.

**Prototyping Model**

The prototype software process model is an iterative and incremental approach to software development. It involves the creation of a working prototype of the software early in the development lifecycle, which is then refined and improved through multiple iterations based on feedback from users and stakeholders. The prototype serves as a tangible representation of the software, allowing users and stakeholders to provide valuable insights and suggestions for improvement.

**The specific milestones associated with the prototype software process model**

**Requirements Gathering**: This milestone involves understanding and documenting the initial set of requirements for the software. It includes identifying the key features and functionalities that the prototype should demonstrate.

**Prototype Development**: In this milestone, the initial prototype of the software is developed. The focus is on creating a basic version of the software that can be demonstrated to users and stakeholders. The prototype may not have full functionality but should showcase the core features.

**Prototype Review**: Once the prototype is developed, it is reviewed by users, stakeholders, and the development team. Feedback is collected to identify areas of improvement and potential changes to the design and functionality.

**Iterative Refinement**: This milestone marks the iterative phase of the process. Based on the feedback received during the prototype review, the development team refines and enhances the prototype, incorporating suggested changes and improvements.

**User Evaluation**: The refined prototype is then evaluated by users to gather their feedback on usability, performance, and functionality. This evaluation helps in identifying any gaps or areas that require further refinement.

**Prototype Enhancement**: The feedback received from the user evaluation is used to enhance the prototype further. Additional features and functionalities may be added, and any issues or concerns raised by users are addressed.

**Final Product Development**: Once the prototype has been sufficiently refined and validated, the development team moves into the final product development phase. The insights gained from the prototype process are used to guide the development of the complete software product.

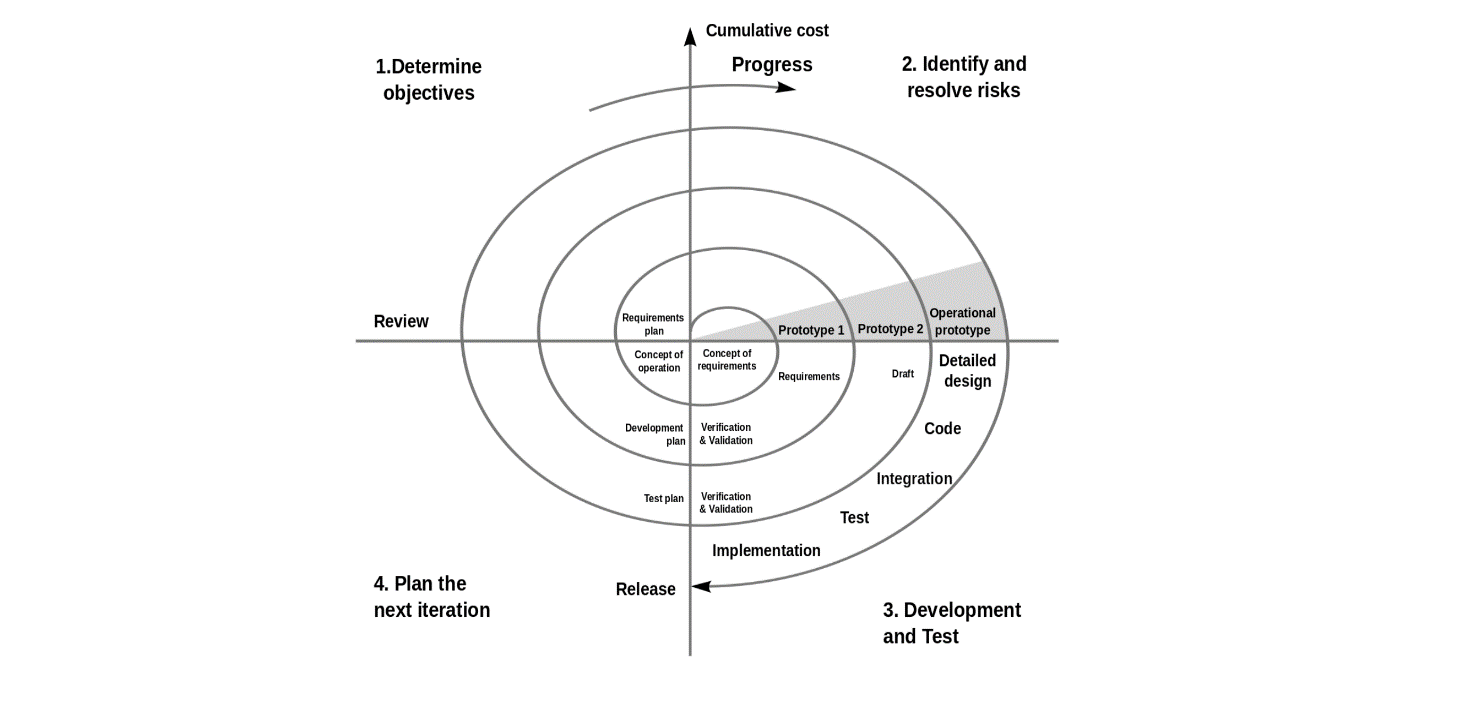
**Spiral Model**

The Spiral model is a software development process model that combines elements of both the waterfall model and iterative development. It emphasizes risk analysis and gradual development, allowing for flexibility and adaptation throughout the project**.** It is a risk driven iterative software process model.

It was designed to include the best features from the waterfall and introduces risk-assessment.

Spiral model has the following phase cycles:

1. Address the highest-risk problem and determine the objective and alternate solutions
2. Evaluate the alternatives and identify the risks involved and possible solutions
3. Develop a solution and verify if it’s acceptable
4. Plan for the next cycle



You develop the concept in the first few cycles, and then it evolves into an implementation. Though this model is great for managing uncertainty, it can be difficult to have stable documentation. The spiral model can be used for projects with **unclear needs** or projects still in research and development.

**Milestones associated with the Spiral model:**

* Project kick-off and objective determination.
* Completion of the first iteration, including requirements analysis, risk assessment, and architectural design.
* Completion of the second iteration, including detailed design, coding, and unit testing.
* Completion of the third iteration, including integration testing and system testing.
* Evaluation of the software increment and risk resolution effectiveness.
* Planning for the next iteration, incorporating feedback and lessons learned.

**The Agile software process model**

The Agile software process model is an iterative and incremental approach to software development that focuses on flexibility, collaboration, and delivering value to customers. Agile methodologies, such as Scrum or Kanban, are widely used in Agile software development.

**Milestones associated with the Agile software process model**

**Project Kick-off**:

* Forming the Agile team, including the Product Owner, Scrum Master (in Scrum), and development team members.
* Defining the project's vision, goals, and objectives.
* Establishing the initial product backlog, which includes a prioritized list of user stories or requirements.

**Sprint Planning**:

* Selecting a set of user stories or backlog items from the product backlog to be implemented in the upcoming sprint.
* Collaborating with the development team to estimate effort, break down user stories into tasks, and create a sprint backlog.
* Defining the sprint goal and identifying the acceptance criteria for each selected user story.

**Sprint Execution**:

* Developing and implementing the selected user stories or backlog items based on the sprint backlog.
* Collaborating daily through stand-up meetings to discuss progress, address challenges, and synchronize team activities.
* Continuous integration of developed features and conducting regular code reviews.

**Sprint Review:**

* Demonstrating the completed work to stakeholders and gathering feedback.
* Collecting insights from stakeholders on the implemented features and incorporating their suggestions for improvements.
* Assessing whether the sprint goals were met and determining whether the product increment is ready for release.

**Sprint Retrospective:**

* Reflecting on the sprint and identifying areas of improvement in the development process, team collaboration, and productivity.
* Discussing what went well, what could be improved, and potential action items for implementing process enhancements.
* Making adjustments to the process or team practices based on the retrospective findings.

**Release and Deployment:**

* Releasing the product increment to users or customers.
* Ensuring that the release meets the desired quality criteria and is ready for deployment.
* Monitoring the deployment process and addressing any issues or bugs that arise.

**Ongoing Iterations:**

* Repeating the sprint cycle with continuous improvements and refinements based on user feedback and changing requirements.
* Conducting regular sprint planning, execution, review, and retrospective activities for each iteration.
* Continuously updating and reprioritizing the product backlog based on evolving customer needs and market demands.

**References**

Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education. (Chapter 2 provides an overview of software process models, including the waterfall model, iterative models, and evolutionary models.)

Sommerville, I. (2016). Software Engineering. Pearson Education Limited. (Chapter 4 covers software process models, including the waterfall model, incremental development, and agile methods.)

Boehm, B. W. (1988). A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes, 11(4), 14-24. (This paper introduces the Spiral Model, which is an iterative and risk-driven software process model.)

IEEE Computer Society. (1991). IEEE Standard Glossary of Software Engineering Terminology (ANSI/IEEE Std 610.12-1990). (This standard provides definitions of various software process models and related terminologies.)

Boehm, B., & Turner, R. (2004). Balancing agility and discipline: evaluating and integrating agile and plan-driven methods. Addison-Wesley Professional. (Chapter 2 discusses software process models and their selection criteria.)