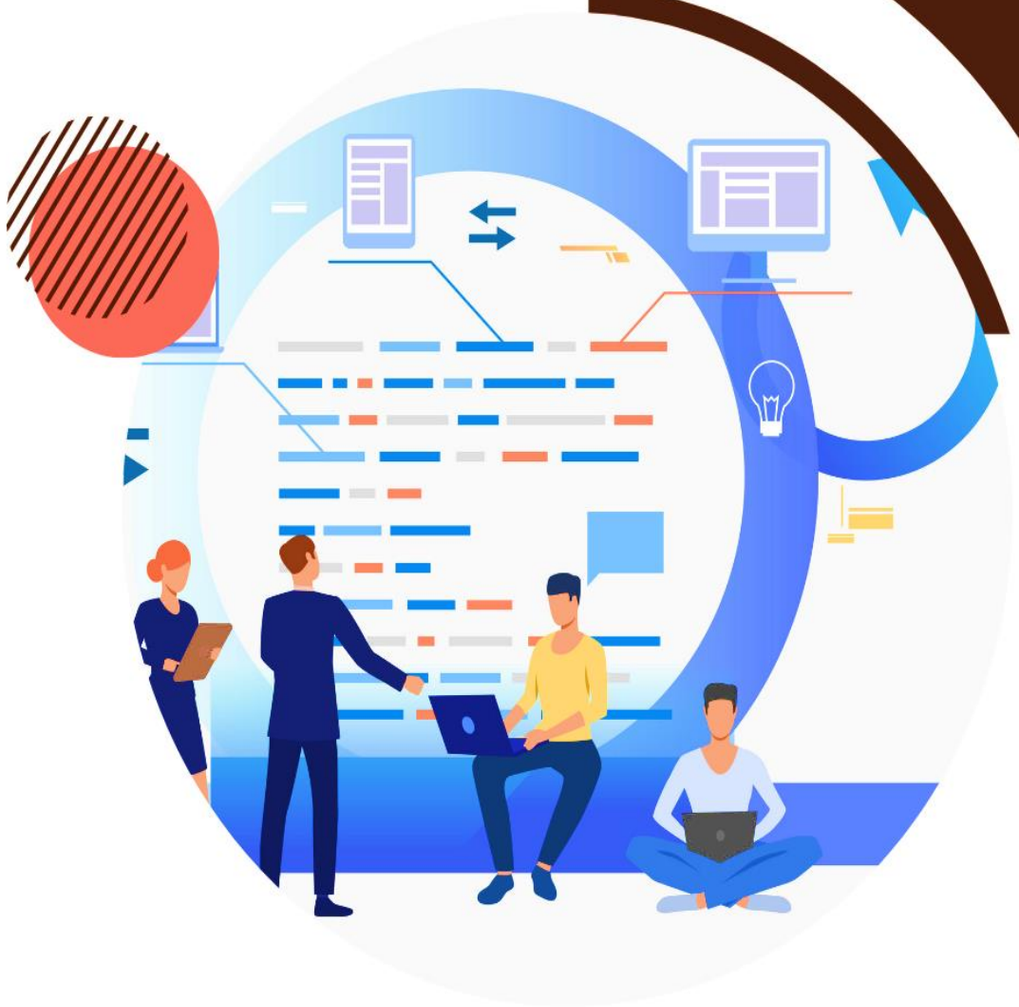
Republic of the Philippines
SORSOGON STATE UNIVERSITY – BULAN CAMPUS
Information and Communication Technology Department



CAPSTONE PROJECT 1

BSIT 3 1-4

Methodology

JESSA P. OSCILLADA, MIT
Instructor

Learning Objectives:

- Determine the IEEE 1074: Standards for Developing Life Cycle Processes
- Identify different kinds of methodology
- fine the systems approach and it's impact on project management
- Determine a Project Management Life Cycle (PMLC) and understand how to

IEEE 1074: Standards for Developing Life Cycle Processes

• The IEEE 1074: Standards for Developing Life Cycle Processes describes the set of activities and processes that are mandatory for the development and maintenance of software.

• The goal of IEEE 1074 is to establish a common framework for developing life cycle models and provide examples for typical situations.

Software Development Methodology

• Software development methodology refers to structured processes involved when working on a project. The goal is to provide a systematic approach to software development.

• A standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

• Describe the tools, techniques and roles in software development.

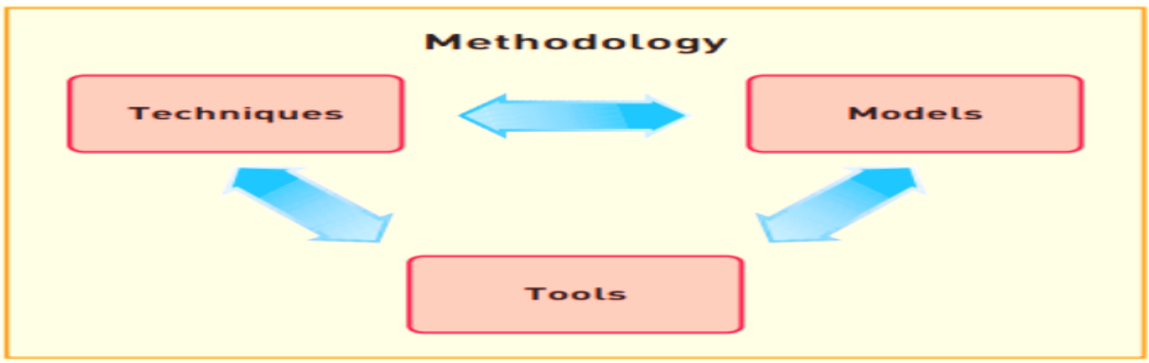


Figure 1. Methodology

• A methodology includes a collection of techniques that are used to complete activities of the software development project.

• The methodology activities include completion of a variety of models.

• Methodology activities are also aided and completed using tools.

MODEL

• A representation of an important aspect of the real world.

• Can be in the form of diagrams and mathematical representations.

• Example: Unified Modeling Language (UML), PERT, Gantt Chart,

DFD, etc.

TOOL

• Software support that helps create models or other components required

in the project.

• Example: DBMS, Project Management Tool, CASE Tool, Drawing Tool, etc.

TECHNIQUES

• A collection of guidelines that help an analyst complete a system development activity.

• Also called as best practices.

• Example: Security Design Technique, Coding Techniques, Strategic Planning Techniques, etc.

Types of Software Methodology

1. Waterfall Model of SDLC
2. Modified Model of (SPIRAL SDLC)
3. V-Shaped Model
4. Agile Methodology
5. Scrum
6. Extreme Programming
7. Feature-Driven
8. Rapid Application Development
9. Rational Unified Process

SYSTEM DEVELOPMENT LICE CYCLE (SDLC)

• The systems development life cycle (SDLC) is the process of determining how an information system (IS) can support business needs, designing the system, building it, and delivering it to users.

• Normally, the SDLC includes all activities needed for the planning, systems analysis, systems design, programming, testing, and user training stages of information systems development, as well as other project management activities that are required to successfully deploy the new information system.

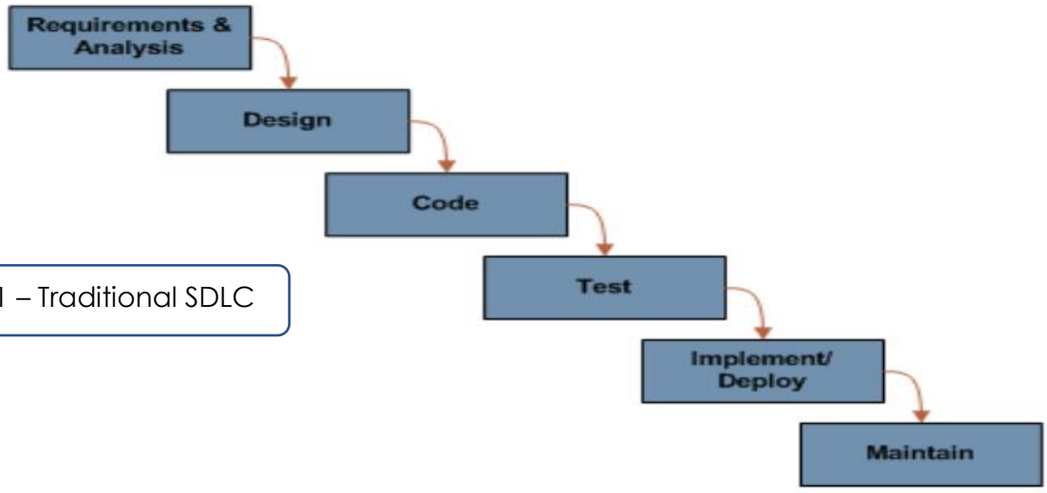
- **WATERFALL SDLC**

• It is also referred to as a linear-sequential life cycle model.

• Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase.

• The key deliverables for each phase are typically very long (often) hundreds of pages in length) and are presented to the project sponsor for approval as the project moves from phase to phase. Once the sponsor approves the work that was conducted for a phase, the phase ends and the next one begins.

Approach 1 – Traditional SDLC



Approach 2

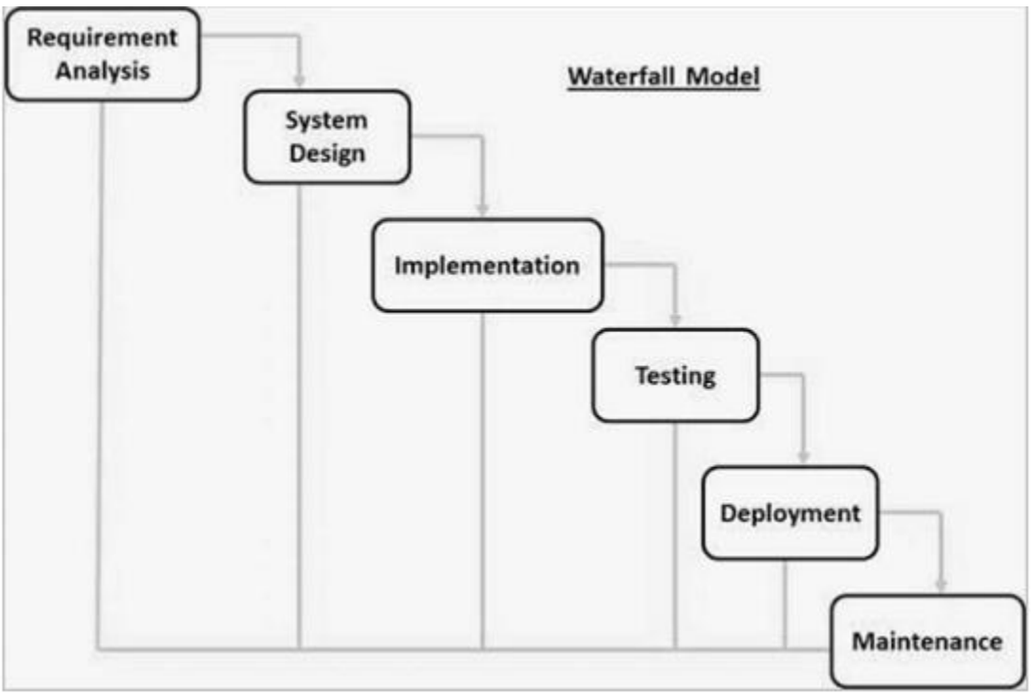


Figure 2. Waterfall Methodology

- **SPIRAL SDLC**
  - Other term is Iterative model the resembles a waterfall model.
  - It implies that the whole process is divided into a particular number of iterations, and during each of them, developers build a limited number of features.
  - The Iterative SDLC model does not require a complete list of requirements before the project starts. The development process may start with the requirements to the functional part, which can be expanded later. The process is repetitive, allowing to make new versions of the product for every cycle.

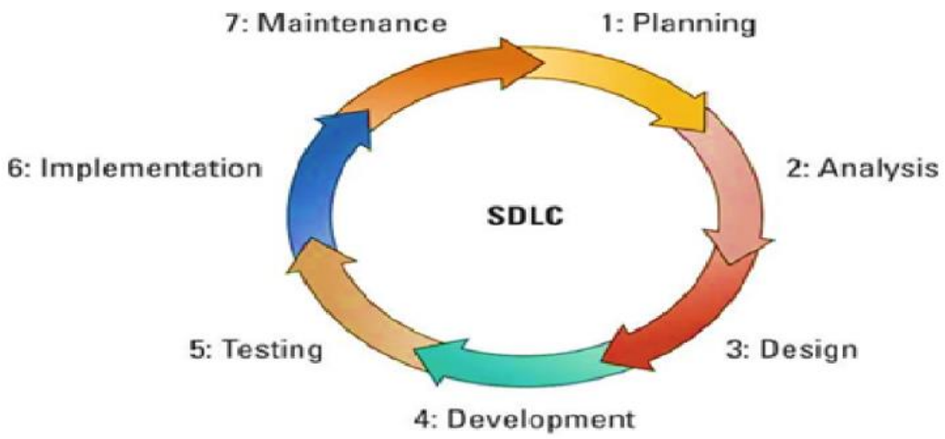


Figure 3. Spiral SDLC

- **V-Shaped Model**

- V-Model is an extension of waterfall model where the execution process flow in a "V-shape".
- V-Model specifies series of phases that executes in sequential, one at a time until the project completes.
- Known as **"Verification and Validation Model"**.
- **Verification:** It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.
- **Validation:** It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

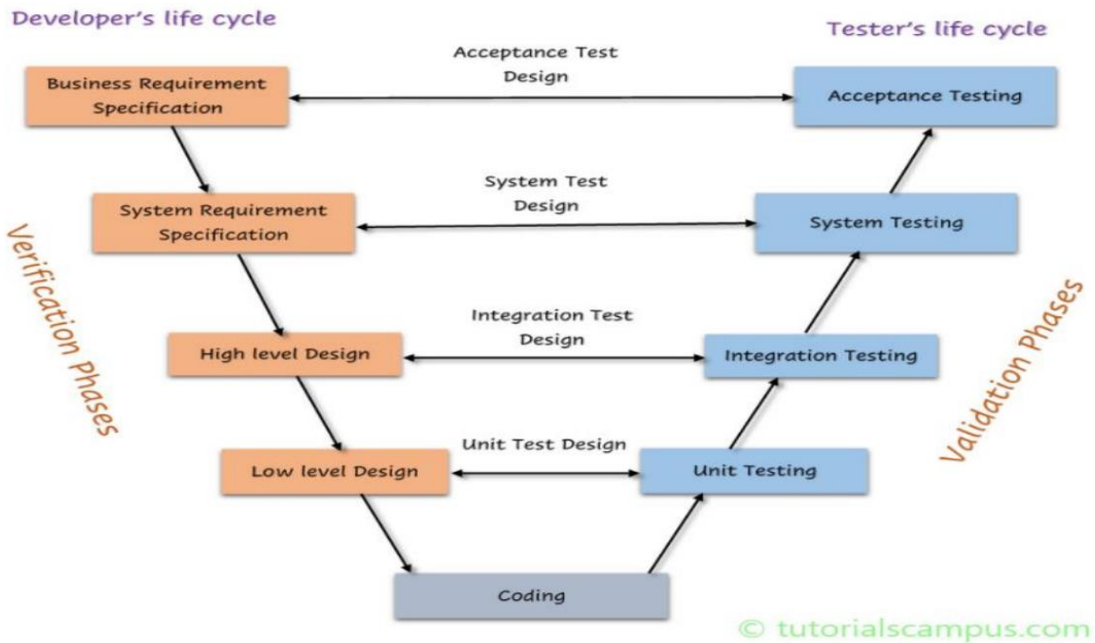


Figure 4. V-Shaped Model

Verification Phase:

- **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.

- **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.

- **Architectural Design(High Level):** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.

- **Module Design(Low Level):** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

Validation Phases:

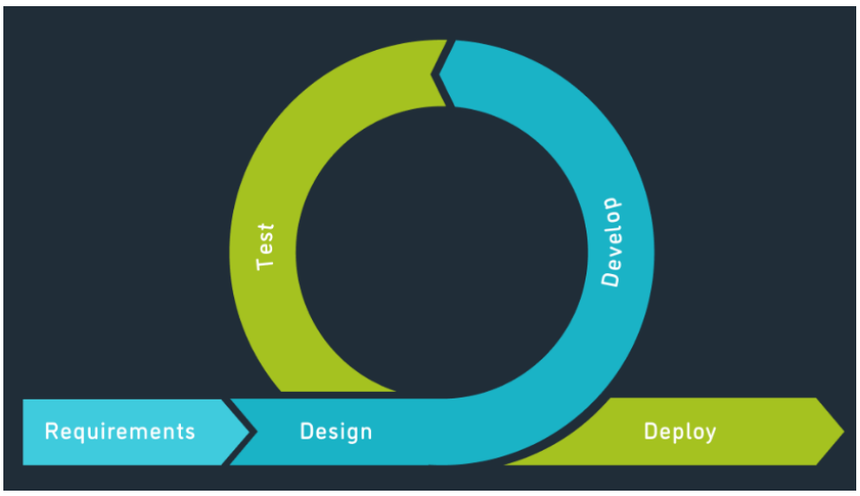
- **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.

- **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.

- **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication. It tests the functional and non-functional requirements of the developed application.

- **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.

- **Agile Methodology**



• Agile is the mainstream methodology used in modern software development, and expands its influence beyond coding into many aspects of product development, from ideation to customer experience.

• Agile methods typically use a spiral model, which represents a series of iterations, or revisions, based on user feedback. As the process continues, the final product gradually evolves.

- Is a way to manage a project by breaking it up into several phases.

- An agile approach requires intense interactivity between developers and individual users, and does not begin with an overall objective. Instead, the agile process determines the end result.

- Proponents of the spiral model believe that this approach reduces risks and speeds up software development.

Examples of Agile Frameworks:

- Scrum
- Extreme Programming
- Feature-Driven
- Kanban
- RAD
- RUP

1. Scrum

- Scrum is responsive to a highly changing, dynamic environment in which users might not know exactly what is needed and might also change priorities frequently.
- Scrum focuses primarily on the team level. It is a type of social engineering that emphasizes individuals more than processes and describes how teams of developers can work together to build software in a series of short mini projects.
- **So, why is it called Scrum?** People often ask, "Is Scrum an acronym for something?" and the answer is no. It is actually inspired by a scrum in the sport of rugby. In rugby, the team comes together in what they call a scrum to work together to move the ball forward. In this context, Scrum is where the team comes together to move the product forward.

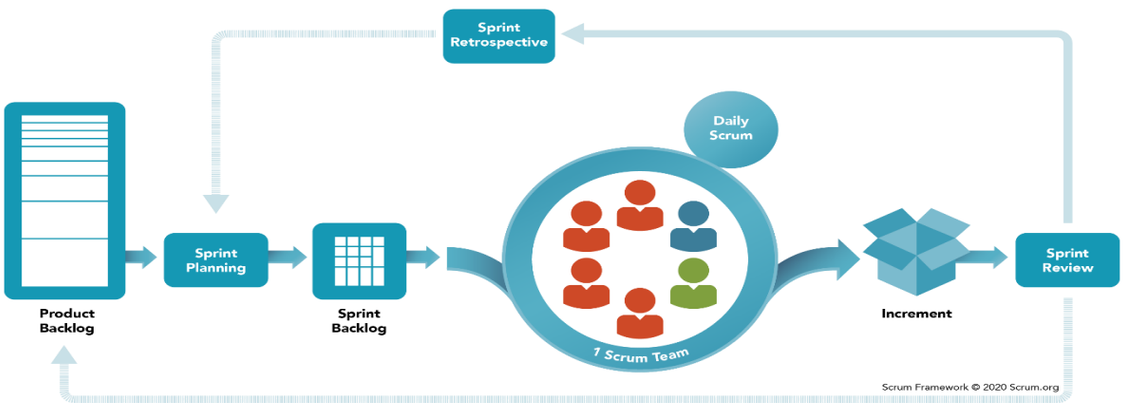


Figure 5. Scrum

- **Product backlog** a prioritized list of user requirements used to choose work to be done in a Scrum project.
- includes user functions (such as use cases), features (such as security), and technology (such as platforms).

The three main organizational elements that affect a Scrum project are the product owner, the Scrum master, and the Scrum team or teams.

- **Product owner** the client stakeholder for whom the system is being built.
- **Scrum master** the person in charge of a Scrum project similar to a project manager.
- **The Scrum team** is a small group of developers—typically five to nine people—who work together to produce the software. For projects that are very large, the work should be partitioned and delegated to smaller teams. If necessary, the Scrum masters from all the teams can coordinate multiple team activities.

Scrum Events
Events that create regularity and minimize other meetings

- **Sprint** - short cycles of one month or less, during which the work is done; the Sprint contains all of the other Scrum events; a new Sprint starts immediately after the conclusion of the previous Sprint
- **Sprint Planning** - event dedicated to planning out the work that will take place during the Sprint
- **Daily Scrum** - event held every day where the Developers inspect the progress toward the Sprint Goal, uncover anything that may be getting in their way and adapt accordingly
- **Sprint Review** - event held at the end of the Sprint where the Scrum Team and key stakeholders review what was accomplished in the Sprint and what has changed in their environment; next, attendees collaborate on what to do next
- **Sprint Retrospective** - the Scrum Team gets together during this event to talk about how the last Sprint went and identify the most helpful changes to improve their effectiveness

Scrum Artifacts
The plans and work which are transparent and can be inspected allowing for future adaptation; Each artifact has its own Commitment which helps the team understand if they are making progress

- **Product Backlog** - an evolving, ordered list of what is needed to improve the product; it is the single source of work undertaken by the Scrum Team
  - Commitment: *Product Goal* - the target the team plans against
- **Sprint Backlog** - a highly visible list of work that is the Developer's plan for the Sprint, which may evolve as they learn
  - Commitment: *Sprint Goal* - the single objective of the Sprint

- **Increments** - small pieces of work that serve as concrete stepping stones toward the Product Goal. You can deliver as often as needed during the Sprint and are not limited to only one release per Sprint.
    - Commitment: *Definition of Done* - the description of what it takes for an Increment to be considered complete

Scrum Practices

- **Sprint** a time-controlled mini-project that implements a specific portion of a system.
- A **Scrum sprint** is a firm period called a time box, with a specific goal or deliverable. At the beginning of a sprint, the team gathers for a one-day planning session.
- After the team has agreed on a goal and has selected items from the backlog list, it begins work. The scope of that sprint is then frozen, and no one can change it—neither the product owner nor any other users.
- If users do find new functions they want to add, they put them on the product backlog list for the next sprint. If team members determine that they can't accomplish everything in their goal, they can reduce the scope for that sprint. However, the time period is kept constant.
- Every day during the sprint, the Scrum master holds a daily Scrum, which is a meeting of all members of the team. The objective is to report progress. The meeting is limited to 15 minutes or some other short time period. The purpose of this meeting is simply to report issues.

2. eXtreme Programming

▪ It is distinguished by its short cycles, incremental planning approach, focus on automated tests written by programmers and customers to monitor the development process.

▪ XP emphasize values such as simplicity, communication, feedback, respect, and courage. Success requires strong commitment to the process, corporate support, and dedicated team members.

▪ Another important concept in XP is that unit tests are designed before code is written. This **test-driven development (TDD)** focuses on end results from the beginning and prevents programmers from straying from their goals. Agile testing relies heavily on automated testing methods.

Extreme Programming is best known for the following:

- **Pair programming** is a technique where two programmers share the same workstation and create software together. One acts as the driver and the other one as the navigator, then they switch roles. When paired, code review can take place instantly, and defects are more likely to be identified and corrected immediately. Pair programming encourages mentorship, knowledge sharing, and learning. And while it

may take more time to produce new code when two developers work on the same task, the resulting code is higher quality with less defects.

- **Unit and functional testing** are emphasized in XP. Tests are to be comprehensive and automated, reducing technical debt and ensuring code can confidently be validated and re-used.

- **Continuous communication** between programmers and stakeholders to gather and act upon their input, feedback, and change requests. XP requires an "extended development team" that may include business managers, customers, and other key stakeholders.

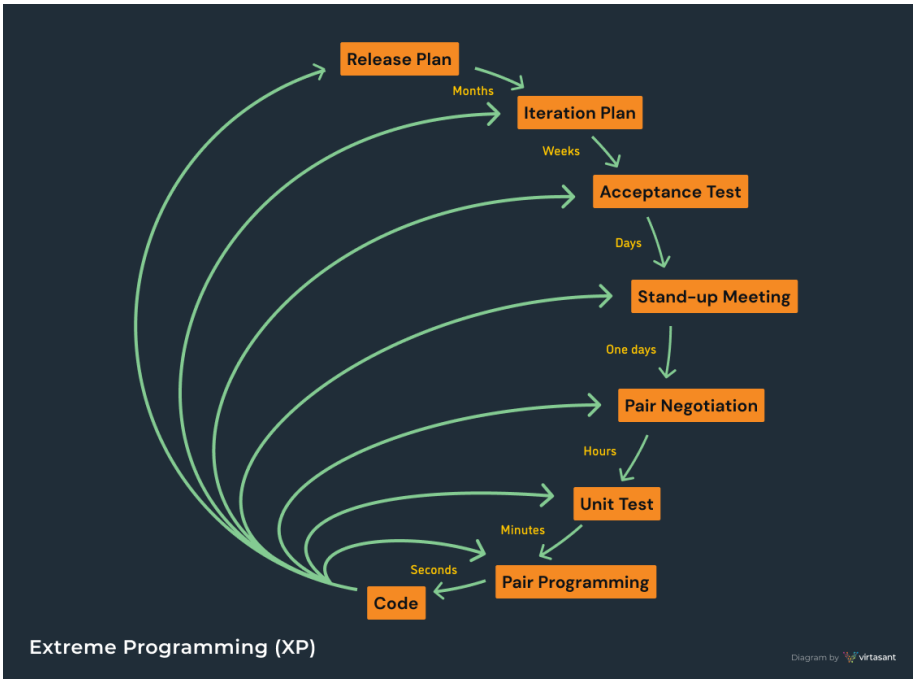


Figure 6. eXtreme Programming

3. Feature Driven

▪ FDD is a model-driven, and short-iteration process that was developed around software engineering best practices including domain object modeling, developing by feature, and code ownership.

▪ Feature-driven development begins with the establishment of an overall model that is expected to result in the feature list.

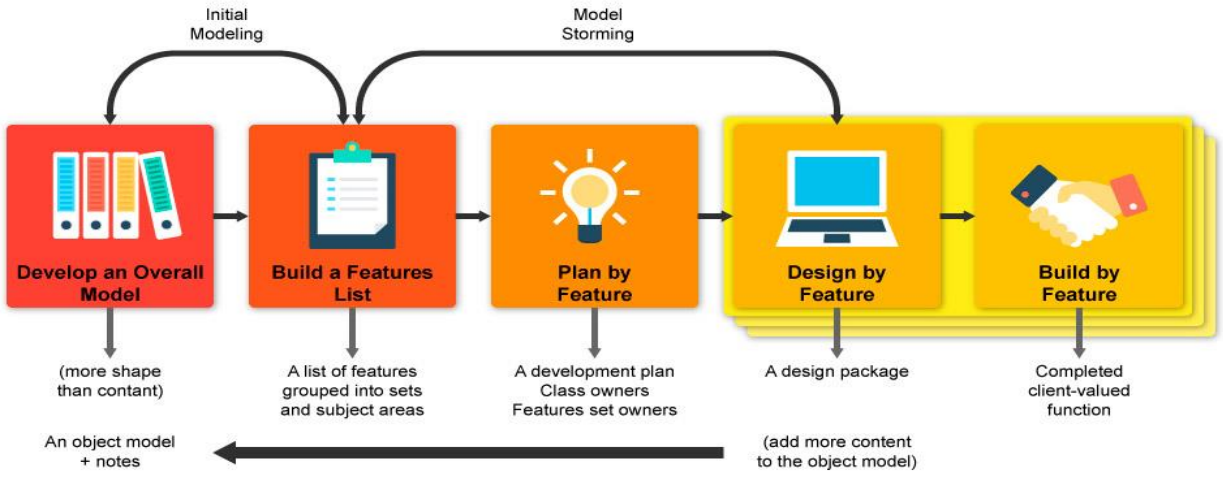


Figure 7. Feature Driven

Step №1. Developing an Overall Model

- In the first stage, the development team members cooperate together to build an object model of the domain problem. The main goal is to propose a model for the domain area. The Chief Architect follows them and provides guidance. Once the teams proposed their own models, one of these models or a merge of models is chosen and it becomes the model created for that domain area. As a result, the team has a clear image of the entire project.

Step №2. Building a Feature List

- After the development team built an object model, then it is time to identify the features that the user or client values. These features are meant to be the building barriers of the project that help the group members to navigate the processes.

Step №3. Planning by the Feature

- The third stage turns around managing the features and the way the development team tends to implement them. As anticipated, it's essential to consider the team workload, risks, as well as other important aspects in order to prevent any kind of complex issues from arising.

Step №4. Designing by the Feature

- Everything planned pretends design. Using the knowledge from the first modeling process, the chief programmer selects all the features that the team should develop next and also identifies the domain classes. After the team starts working on the project, the domain expert analyzes and designs a solution to each feature.

Step №5. Building by the Feature

- The last step is to put all the necessary items into action in order to support the design. In other words, once your team developed, tested and inspected the code, it is time to start developing the software.

4. Kanban

- is a method that helps businesses plan and organizes project activities visually by prioritizing tasks on boards.
- is an agile method that aims at continuous improvement, flexibility in task management, and enhanced workflow. With this illustrative approach, the progress of the whole project can be easily understood in a glance.

Kanban board

- A Kanban board is a project visualization tool that helps team management work effortlessly. Kanban boards are usually represented as interactive panels with virtual sticky notes that can be moved around easily to organize tasks and to-do lists.
- Kanban is a scheduling system framework for the Agile-eque Lean methodology. It doesn't have its roots in software development, but synergizes very well with Agile and has become a staple of Agile teams.
- Kanban got its start in lean manufacturing, where Toyota applied the same "just in time" principles that supermarkets use to manage inventory stock levels based on customer demand. Kanban, meaning signboard in Japanese, uses cards to track and support the production system by visually showing the steps within the process and how long each step is taking using cards.

Kanban board indicates

- the current tasks that are being performed
- the tasks to do in the future
- the tasks that are completed

A few examples of popular Kanban productivity apps:

- Trello
- Jira
- Proofhub
- Zoho Projects
- ZenHubs

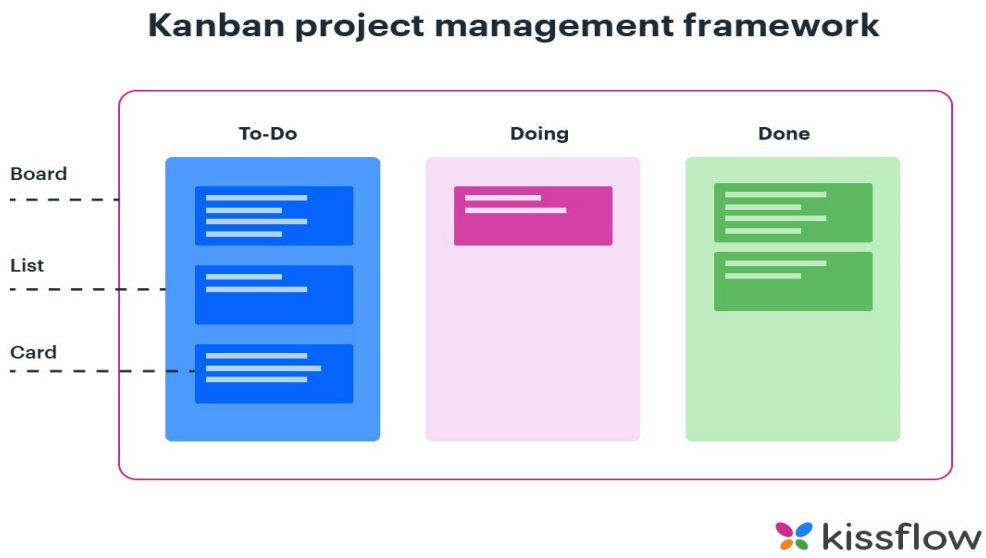
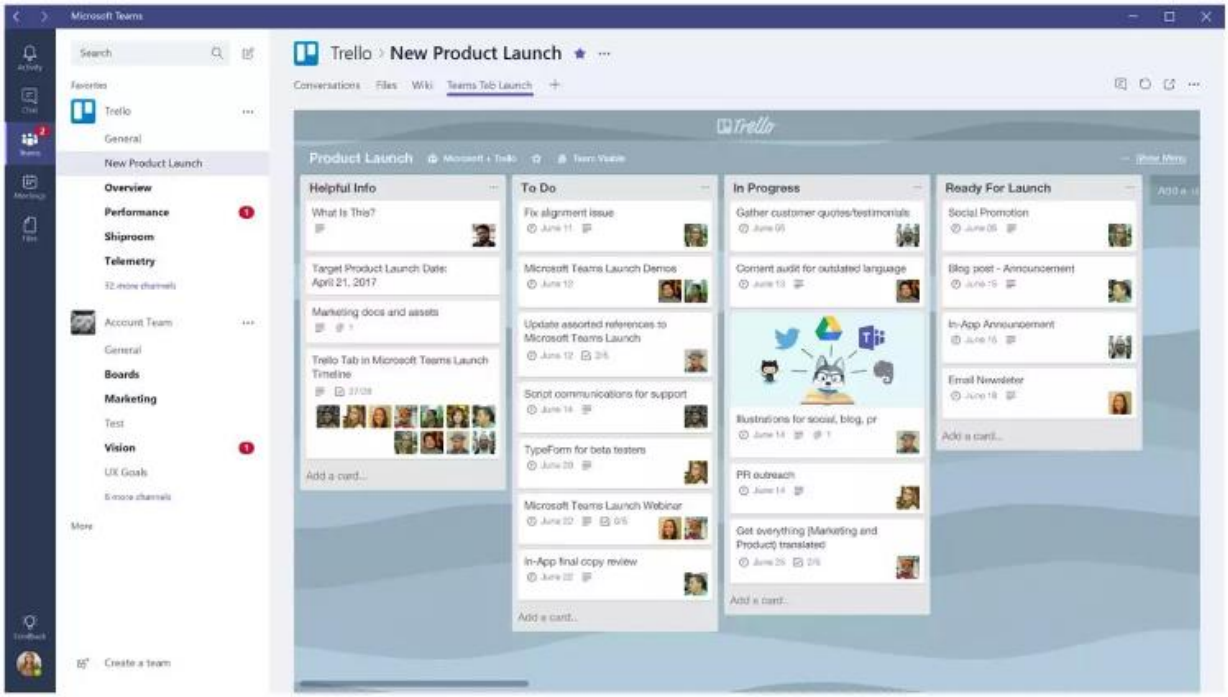


Figure 8. Kanban

- As soon as a task is completed, the team can take the next item from the pipeline. Thus, the development process offers more flexibility in planning, faster turnaround, clear objectives, and transparency.
- No standard procedures within the process, as well as the fixed iterations, are required in Kanban, as opposed to Scrum. The project development is based on the workflow visualization through a **Kanban board**, usually represented by sticky notes and whiteboards or online tools like Trello.



- Trello automates and digitalizes Kanban. Due to the succinct information about a work item each Kanban card contains, everyone in the team knows who is responsible for the item, what each person's task is, when it's supposed to be finished, etc. Team members can also leave comments, attach screenshots, documents, or links to provide more details.
- Teams using Kanban tools work in a cooperative manner. The ability to track progress helps coworkers understand everyone's personal input in achieving the common goal, resulting in a focus on completing the task well and on time.

When to Use Kanban

- Using Kanban, teams can do small releases and adapt to changing priorities. Unlike Scrum, there are no Sprints with their predefined goals. Kanban is focused on doing small pieces of work as they come up. For example, if testers find errors in the product, developers try to fix them right away. Kanban, for instance, works well after the main release of the product and suits update and maintenance purposes.
- Companies like Spotify and Wooga (a leading mobile games development company) have been using this approach successfully over the years. Yet, 8 percent of organizations combine Scrum with Kanban techniques, using so-called **Scrumban** rather than the original frameworks.

5. RAPID APPLICATION DEVELOPMENT

- is a software development methodology that focuses on building applications in a very short amount of time.
- It is a high speed adaptation of the linear sequential model in which rapid development is achieved by using component based construction.

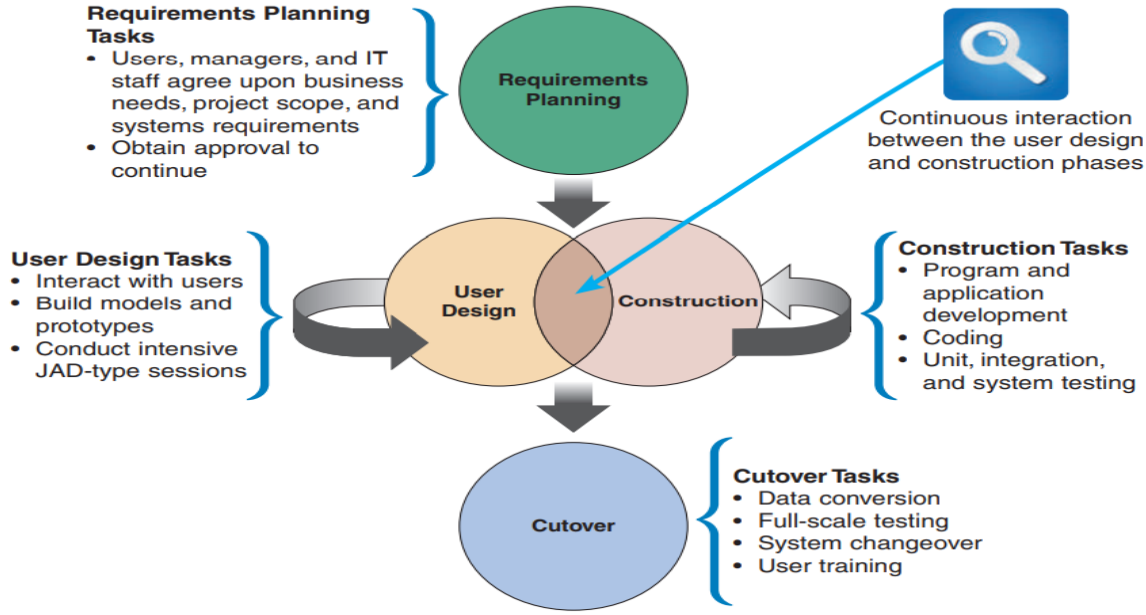


Figure 9. Rapid Application Development

Phase 1. Requirement Planning

The requirements planning phase combines elements of the systems planning and systems analysis phases of the SDLC. Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements. The requirements planning phase ends when the team agrees on the key issues and obtains management authorization to continue.

A basic breakdown of this stage involves:

- Researching the current problem
- Defining the requirements for the project
- Finalizing the requirements with each stakeholder's approval

Phase 2. User Design

During the user design phase, users interact with systems analysts and develop models and prototypes that represent all system processes, outputs, and inputs. User design is a continuous, interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs.

Phase 3. Construction

The construction phase focuses on program and application development tasks similar to the SDLC. In RAD, however, users continue to participate and still can suggest changes or improvements as actual screens or reports are developed.

Phase 4. Cutover

This phase takes the prototypes and beta systems from the design phase and converts them into the working model. Once the product is eventually approved, developers put some finishing touches in the form of testing, conversion, interface, or user training. Once the product is properly assessed for factors like stability and longevity, it is ready to be delivered.

6. RATIONAL UNIFIED PROCESS

An object-oriented systems development methodology. RUP establishes **four phases of development**: **inception, elaboration, construction, and transition**. Each phase is organized into a number of separate iterations.

It is also known as the Unified Process Model. It is created by Rational corporation and is designed and documented using UML **(Unified Modeling Language).**

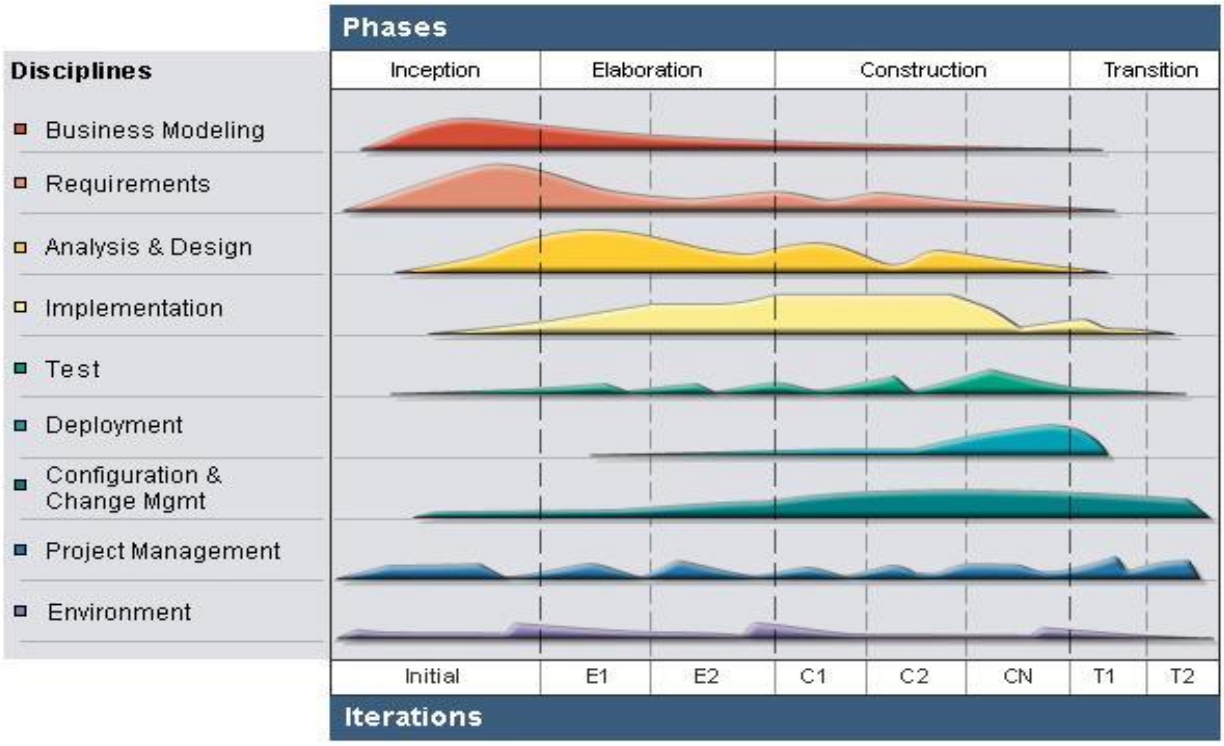


Figure 10. Rational Unified Process

Unified Process Systems Development Methodology Disciplines

- **Business Modeling**
  - Understand the business environment
  - Create the system vision
  - Create business models
- **Requirements**
  - Gather detailed information
  - Define functional requirements
  - Define non-functional requirements
  - Prioritize requirements
  - Develop user interface dialogs
  - Evaluate requirements with users
- **Design**
  - Design the support service architecture and deployment environment
  - Design the software architecture

- Design use case realization
- Design the database
- Design the system and user interface
- Design the system security and controls

- **Implementation**
  - Build software components
  - Acquire software components
  - Integrate software components
- **Testing**
  - Define and conduct unit test
  - Define and conduct integration test
  - Define and conduct usability test
  - Define and conduct user acceptance test
- **Deployment**
  - Acquire hardware and system software
  - Package and install components
  - Train users
  - Convert and initialize data
- **Project Management**
  - Finalize the system and project scope
  - Develop the project and iteration schedule
  - Identify project risks and confirm the project's feasibility
  - Monitoring and controlling the project's plans
- **Configuration and Change Management**
  - Develop change control procedures
  - Manage models and software components
- **Environment**
  - Select and configure the development tools
  - Tailor the Unified Process development process

- Provide technical support services

These four building blocks are:

- **(Roles) the 'Who':** It shows who are the responsibilities for developing the software product. It may be an individual or a group of individuals together as a team who work on it.
- **(Work Products) the 'What':** It indicates what will be produced. That shows the behavior and type of software product.
- **(Workflows) the 'When':** It represents the flowchart of activities in order to produce a software product.
- **(Tasks) the 'How':** It describes how the development will take place, i.e. a unit of work assigned to a Role to perform and that provides a meaningful result.

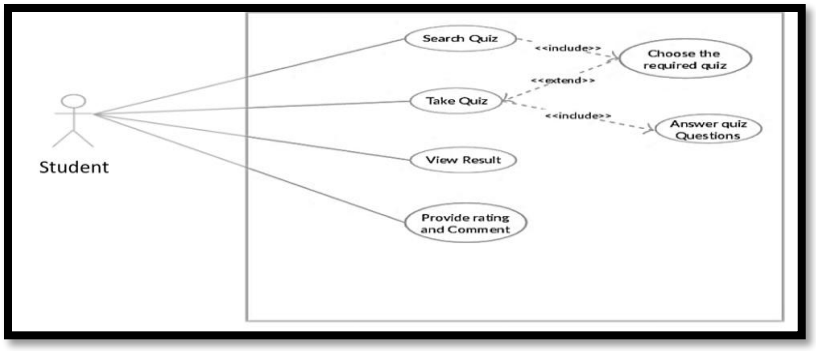
Phases Involved in the Rational Unified Process

It consists of four phases to complete the RUP process, and each phase having a specific purpose.

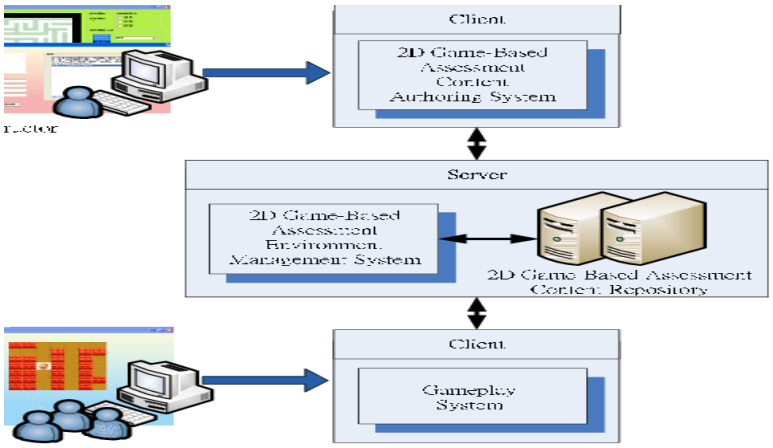
- Inception Phase
- Elaboration Phase
- Construction Phase
- Transition Phase

1. Inception Phase

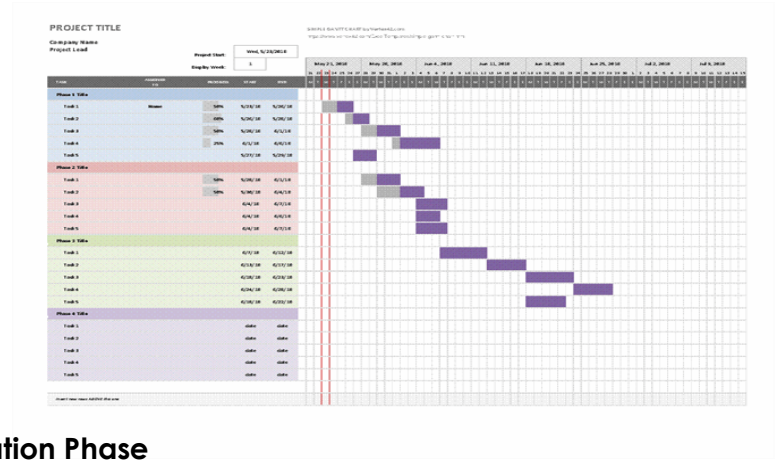
- In the inception phase, analysts define the scope, determine the feasibility of the project, understand user requirements, and prepare a software development plan.
- Requirement understanding.
- Estimated Cost
- What are the major users (actors) and what will the system do for them? Simplified use-case model.



- What could be an architecture for the system? Tentative Architecture.



- What is the plan and how much it will cost?
- Main risk identified, rough planning.



2. Elaboration Phase

- During this phase, to analyze the project's requirements and necessary architecture, i.e. to review the problems, develop the project plan and architect, and eliminate the high-risk elements from the project.

- Refine plan of activities and estimates.
- Almost complete use-case model.
- It provides a full model of the project with functional and non-functional requirements.

3. Construction Phase

- This is the third phase of the development process.
- During this phase, the project is developed and completed. Here all the features are developed and integrated into the product, i.e. the software is designed, written, and tested successfully.
- The development product will be a deployable product. It measures the completeness of the product.

4. **Transition Phase**

- Final project is released to public.
- Transit the project from development into production.
- Update project documentation.
- Beta testing is conducted.
- User Training

References:

Tilley, S., & Rosenblatt H. (2016). Systems Analysis and Design, 11th edition. Cengage Learning, Inc. 20 Channel Center Street Boston, MA 02210 USA.

Dennis, A., Wixom, B., & Tegarden, D. (2015). System Analysis & Design an Object-Oriented Approach with UML, 5th Edition, Wiley Publishing.

J. W. Satzinger, R. B. Jackson & S. D. Burd (2015). Systems Analysis and Design in a Changing World, 7th edition. Cengage Learning, Inc. 20 Channel Center Street Boston, MA 02210 USA

SDLC - Waterfall Model. Retrieved from: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

V-Model. Retrieved from: https://www.tutorialscampus.com/sdlc/v-model.htm

Scrum Framework(2020). Retrieved from: https://www.scrum.org/resources/what-is-scrum

Feature-driven. Retrieved from: https://aist.global/en/use-a-feature-driven-development.

Kissflow (2022). 5 Phases of Project Management – A Complete Breakdown. Retrieved from: https://kissflow.com/project/five-phases-of-project-management/

Coursera (2022). 4 Phases of the Project Management Lifecycle Explained. Retrieved from: https://www.coursera.org/articles/project-management-lifecycle

Digite (2022). What is Project Management Life Cycle? And it's 5 Phases. Retrieved from: https://www.digite.com/project-management/project-management-life-cycle/