**Software Engineering**

**Assignment 1**

**Submission By**

**Team Alchemist (Team 15)**

**Team Members:**

* **Swapnil Narayan Supekar**

**(200940381118)**

* **Swapnil Uttam Andhale**

**(200940381119)**

* **Swapnil Valwe**

**(200940581034)**

* **Tanuja Dnyaneshwar Patil**

**(200940581035)**

* **Tejal Ramrao Desale**

**(200940381121)**

* **Tejas Gopal Rane**

**(200940381122)**

* **Udbhav Garg**

**(200940581036)**

* **Udit Dilip Nemade**

**(200940581037)**

* **Udit Dilip Nemade**

**(200940581037)**

* **Umesh Sunil Chaudhari**

**(200940581036)**

* **Unnati Tappe**

**(200940381124)**

**Aim: To study Software Development Life Cycle (SDLC) Models**

What is SDLC?

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates



A typical Software Development Life Cycle consists of the following stages

Stage 1: Planning and Requirement Analysis

### Stage 2: Defining Requirements

### Stage 3: Designing the Product Architecture

### Stage 4: Building or Developing the Product

### Stage 5: Testing the Product

### Stage 6: Deployment in the Market and Maintenance

There are various software development life cycle models defined and designed which are followed during the software development process.

Listed below are most popular and important models:

1. Waterfall Model
2. Incremental Model
3. Rapid Application Development
4. Spiral Model
5. Agile Model
6. ‘V’ shaped model

**WATERFALL MODEL**

**Introduction**

The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion.

This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases.

Six stages in water fall model are as follows:

1. **Requirements**: In this stage, the development team methodically gathers project requirements — from the customer’s business context to the product performance level — and captures the results in a requirements specification. Well-documented requirements serve a foundation and guarantee a smooth software development process.

2. **Analysis**: The team further analyzes functional and nonfunctional requirements, outlines the project scope and constraints, identifies the stakeholders, and develops the business logic. The phase is critical for obtaining a clear vision of the end product.

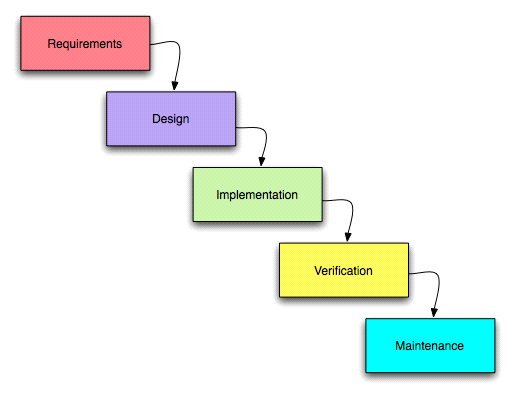
3. **Design**: Using the output from the previous stage, the team proceeds to work on the system’s design, including the software and hardware architecture, database tables, and user interface mockups. The resulting design specification describes how the project will be executed from a technical standpoint.

4. **Coding**: Once the design is approved, the engineers commence the actual coding according to the project specifications. The coding process runs smoothly provided that the project design phase has been properly executed.

5. **Testing**: During this stage, the testing and quality control team performs all testing activities, including functional, system, and acceptance testing to find bugs, errors, or flaws. Testers also make sure that the product is built as per the requirements specification and that it functions as required. Any bugs found in the software mean that the development team needs to go back to the coding stage to fix them.

6. **Operations**: Sometimes referred to as the implementation phase, this is the final stage of the software development life cycle that covers the software deployment to a live environment. It also includes maintenance and subsequent support of the product to ensure it is functioning properly at all times.

**PHASE DIAGRAM**



**Advantages:**

* A logical structure that is easy to understand and to manage;
* Each stage has clearly defined deliverables to review and approve;
* The requirements are documented in great detail and remain unchanged throughout the process;
* Thorough documentation facilitates the onboarding process for new developers;
* The model lends itself well to milestone-focused development process by providing a clear timeline for the project;
* Extensive planning ensures more predictable end results in terms of budget and scope.

**Disadvantages**:

* Lack of flexibility — the linear progression of stages makes going back a step difficult;
* A mistake in the requirements and/or design is not found until the testing phase, which often increases the cost of defect elimination;
* A working product is produced late in the development life cycle;
* The model may be a poor choice for time-sensitive projects or those with no well-defined requirements;
* A strong emphasis on control makes the model unfit for high-uncertainty and high-risk environments.

**Real-Time Example:**

For big projects and in crucial systems such as civil or medical.

Banking systems, you cannot write to code without planning. One error in customer transaction, you could go in big trouble.

**Incremental Model**

**Introduction**

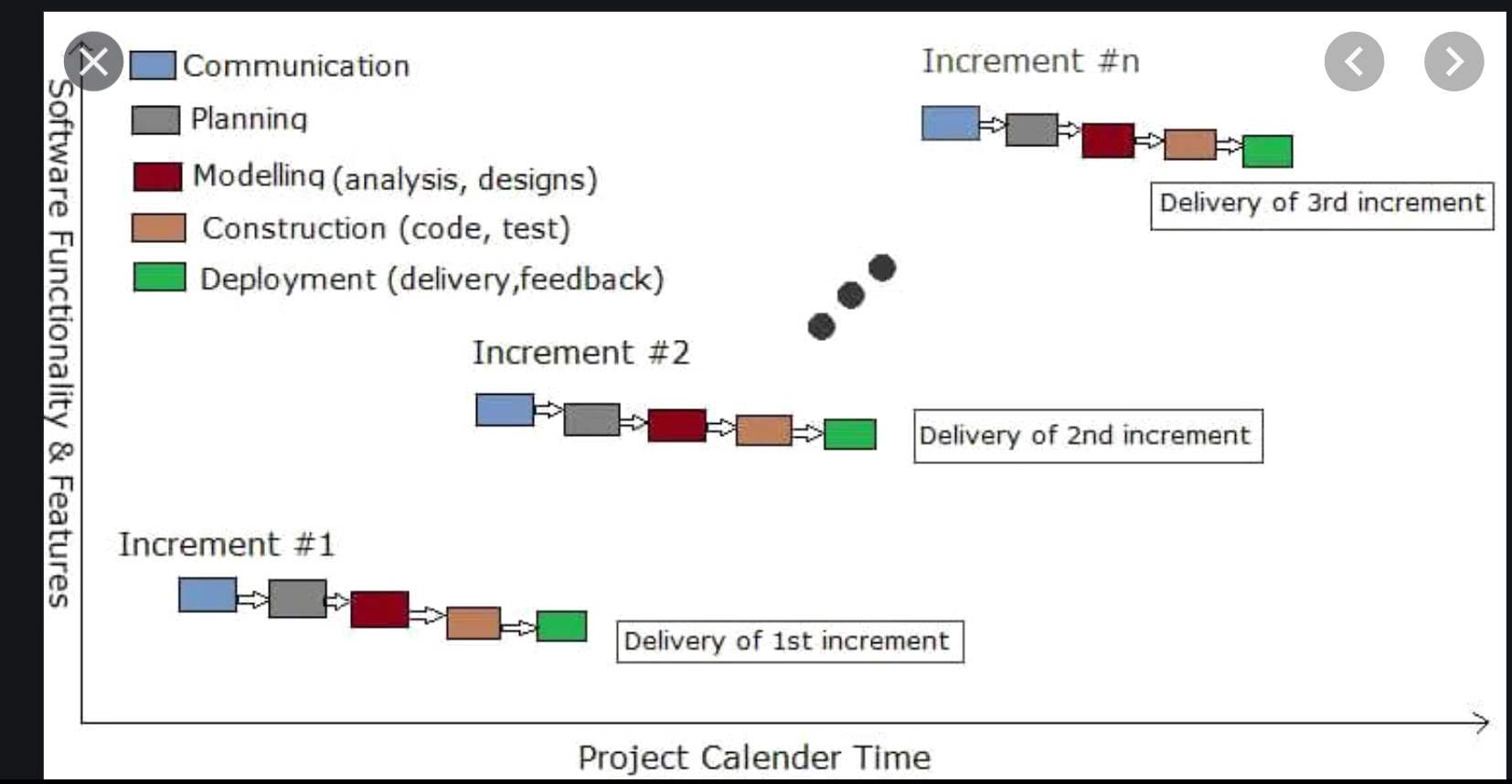
Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. The first increment is often a core product where the basic requirements are addressed and supplementary features are added in the next increments.

**Purpose**

The main purpose of the Incremental model is that it divides the software development into submodules and each submodule is developed by following the software development life cycle process SDLC like Analysis, Design, Code, and Test.

By doing this model make sure that we are not missing any objective that is expected from the end of the software even though how minor objective it can be. Thus we are achieving 100% objective of the software with this model also since we are testing aggressively after each stage we are making sure of the end software is defect-free and also each stage is compatible with previously developed and future developing stages.

**PHASE DIAGRAM**

**Advantages**

* Since the object will be divided into incremental stages, it will be made sure that 100% requirements are achieved and 100% objective of the software.
* Since there is testing at each incremental phase there will be multiple testing for the software and more the testing better the result and fewer defects.
* By adopting this approach, we can lower the initial delivery cost.
* This model is flexible and incurs lest cost when there is a change in the requirement or the scope.
* Effort will be valued and sudden changes in the requirement can be prevented.
* Compared to the other model this model is tend to be cheaper on the pockets of the user.
* By following this models errors can be identified quiet easily.

**Disadvantage**

* It requires a good planning designing
* Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle
* Each iteration phase is rigid and does not overlap each other
* Rectifying a problem in one unit requires correction in all the units and consumes a lot of time

**Real Time Example**



In the diagram above when we work incrementally we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it’s complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise, in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

**RAD (Rapid Application Development) Model**

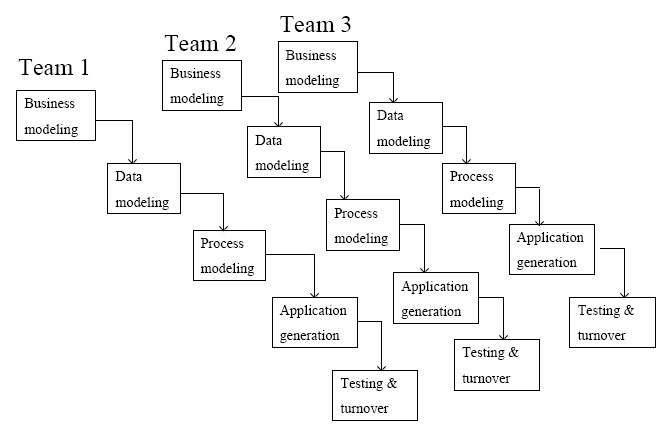
**Introduction:**

Rapid application development (RAD) describes a method of software development which heavily emphasizes rapid prototyping and iterative delivery. The RAD model is, therefore, a sharp alternative to the typical waterfall development model, which often focuses largely on planning and sequential design practices. First introduced in 1991 in by James Martin, rapid application development has become one of the most popular and powerful development method.

**Phases in RAD:**

1. Business modelling: The information flow is identified between different business functions.
2. Data modelling: Information collected from business modelling is used to define data objects that are required for the business.
3. Process modelling: Data objects defined in data modelling are converted to establish the business information flow to achieve some specific business objective process descriptions for adding, deleting, modifying data objects that are given.
4. Application generation: The actual system is created and coding is done by using automation tools. This converts the overall concept, process and related information into actual desired output. This output is called a prototype as it’s still half-baked.
5. Testing and turnover: The overall testing cycle time is reduced in the RAD model as the prototypes are independently tested during every cycle.

**PHASE DIAGRAM**



**Advantages:**

• Reduced development time.

• Increases reusability of components

• Quick initial reviews occur

• Encourages customer feedback

• Integration from very beginning solves a lot of integration issues.

**Disadvantages:**

• For large but scalable projects RAD requires sufficient human resources.

• Projects fail if developers and customers are not committed in a much shortened time-frame.

• Problematic if system cannot be modularized.

• Not appropriate when technical risks are high (heavy use of new technology).

**Real time Examples:**

#### **When You’ve Got the Budget**

Compared to other development models, rapid application development is relatively inexpensive, but there are some instances where RAD can be expensive. Hiring talented staff means you’ll need to give them appropriate salaries. The bright side is, if you’ve got the staff, you can get the idea from concept to end product a lot quicker than other models.

#### **When You Need a Project Done Quickly**

If you’ve got a tight deadline, rapid application development is a best bet. If you’re under pressure to deliver something that works, then opting for a RAD platform can be the best choice. If you don’t have the time to go through a long requirement planning and design phase, then rapid application development software is your best bet. Rapid application development takes an on-the-fly approach, which makes sense for quick development which can change direction on a dime.

**Spiral Model**

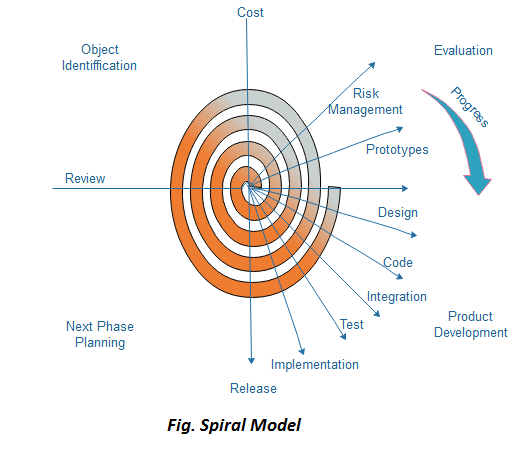
**Introduction**

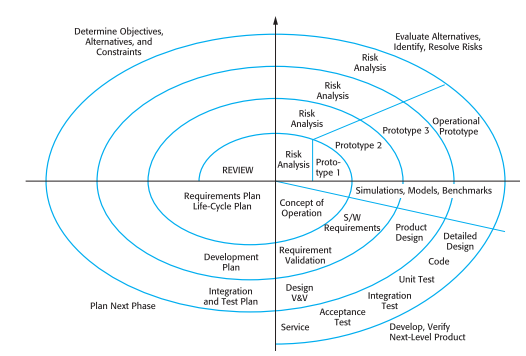
It is a risk-driven software process framework (the spiral model) was proposed by Boehm (1988). Here, the software process is represented as a spiral, rather than a sequence of activities with some backtracking from one activity to another. The spiral model is an evolutionary software process model that combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral. It provides the potential for rapid development of increasingly more complete versions of the software.

**Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project.

Each loop in the spiral represents a phase of the software process. Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design, and so on.

**PHASE DIAGRAM**





**Design and Purpose**

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

1. **Identification and Objectives**

* This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase. Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase.
* This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.
* Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.
* A detailed management plan is drawn up. Project risks are identified. Alternative strategies, depending on these risks, may be planned.

1. **Risk Analysis and Reduction**

* During the second quadrant all the possible solutions are evaluated alternatives based on the goals and constraints to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
* Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.
* Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

1. **Development and Validation**

* The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.
* **Prototyping Model** also support risk handling, but the risks must be identified completely before the start of the development work of the project. But in real life project risk may occur after the development work starts, in that case, we cannot use Prototyping Model.
* In each phase of the Spiral Model, the features of the product dated and analyzed and the risks at that point of time are identified and are resolved through prototyping. Thus, this model is much more flexible compared to other SDLC models.
* Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

**Design**

* The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

**Development and Testing**

* The Product is Developed and tested in this phase.

1. **Planning and Implementation**

* Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.
* The product is Deployed and Customer Feedback is taken into consideration.
* The Maintenance of the product and further enhancement can also be done in this phase.

**Purpose**

## When to use Spiral Model?

* When deliverance is required to be frequent.
* When the project is large and high budget where costs and risk evaluation is important.
* When requirements are unclear and complex
* When changes may require at any time
* When there is a budget constraint and risk evaluation is important.
* For medium to high-risk projects.
* Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
* Customer is not sure of their requirements which is usually the case.
* Requirements are complex and need evaluation to get clarity.
* New product line which should be released in phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.

**Advantages**

The advantages of the Spiral SDLC Model are as follows −

* Change requests in the Requirements at later phase can be incorporated accurately by using this model.
* Cost Estimation becomes easy.
* Allows extensive use of prototypes.
* Requirements can be captured more accurately.
* Users see the system early.
* Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.
* Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product

**Disadvantages**

The disadvantages of the Spiral SDLC Model are as follows −

* Management is more complex.
* End of the project may not be known early.
* Not suitable for small or low risk projects as it could be expensive for small projects.
* Process is complex than other SDLC models.
* Spiral may go on indefinitely.
* The successful completion of the project is highly much dependent on Risk Analysis which requires Specific expertise.
* Large number of intermediate stages requires excessive documentation.

**Real-time Examples**

* One of the most interesting things about the SDLC Spiral model is that Microsoft used it to develop early versions of Windows. The Gantt chart software was made using the model as well. Thus, there are no surprise here — big, high risk projects, which are also aimed at a wide audience, choose Spiral Model.
* Game development is another industry that uses the Spiral model. As we discussed above, the model allows for creating prototyping carefully and quickly. As the gaming industry heavily relies on early game versions, Spiral becomes a solid option. With the model, game development companies can get feedback from their customers really fast and develop a playable that would evolve into equally playable games faster.

**Agile Model**

**Introduction**

Agile is a process by which a team can manage a project by breaking it up into several stages and involving constant collaboration with stakeholders and continuous improvement and iteration at every stage. This clarifies the customer's expectations to the project team.  
  
**Agile Methodology**

Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality.

**Purpose of Model**

The overall goal of each Agile method is to adapt to change and deliver working software as quickly as possible. However, each methodology has slight variations in the way it defines the phases of software development. Furthermore, even though the goal is the same, each team’s process flow may vary depending on the specific project or situation. As an example, the full Agile software development lifecycle includes the concept, inception, construction, release, production, and retirement phases.

**The four values of the Agile Manifesto**

The Agile Manifesto consists of four key values:

* Individuals and interactions over processes and tools.
* Working software over comprehensive documentation.
* Customer collaboration over contract negotiation.
* Responding to change over following a plan.

**The Agile Lifecycle**

There are a variety of Agile software development (or system development) methodologies, including, but not limited to:

**It is a family of iterative , incremental methods.**  
  
1.Disciplined Agile Delivery (DAD)  
2.Adaptive Software Development  
3.Agile Modeling  
4.Kanban  
5.Scrum  
6.Scrumban  
7.Extreme Programming (XP)  
8.Dynamic Systems Development (DSDM)  
9.Feature Driven Development  
10.Lean Software Development

**A typical iteration process flow can be visualized as follows:**  
  
**Requirements**

Define the requirements for the iteration based on the product backlog, sprint backlog, customer and stakeholder feedback

**Development**

Design and develop software based on defined requirements

**Testing**

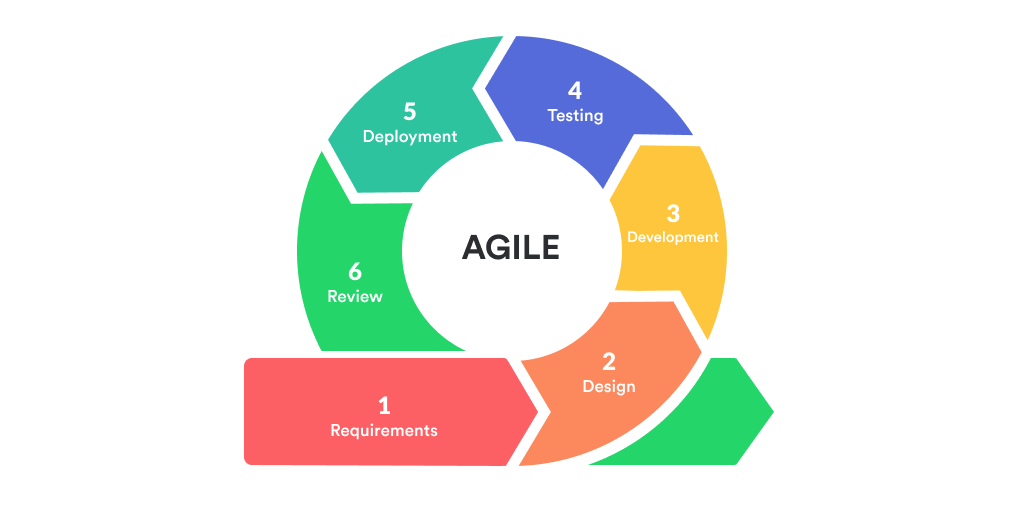
QA (Quality Assurance) testing, internal and external training, documentation development

**Delivery**

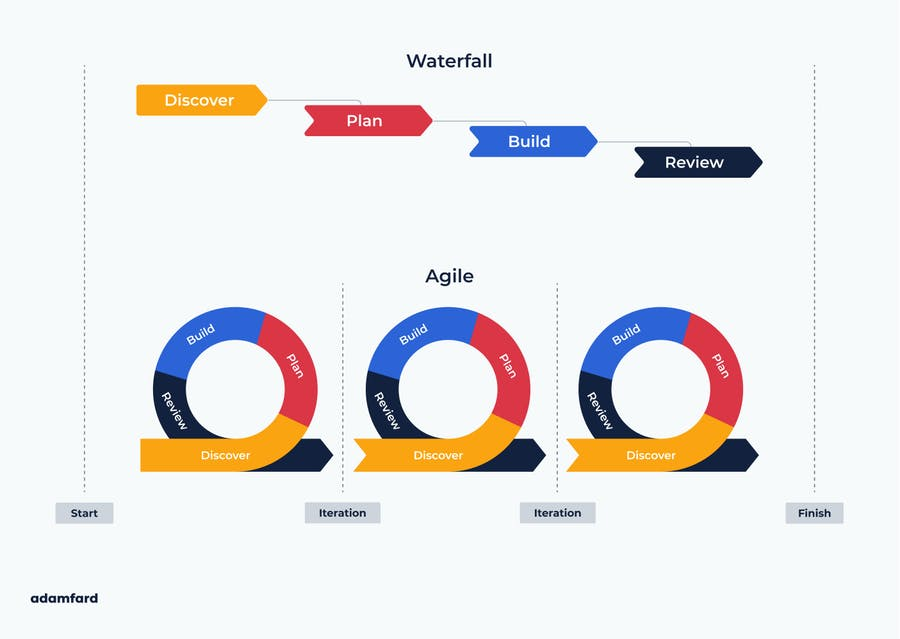
Integrate and deliver the working iteration into production

**Feedback**

Accept customer and stakeholder feedback and work it into the requirements of the next iteration

**Phase Diagram**

**Iterative Approach**

****

## **Advantages**

* Customer satisfaction is rapid, continuous development and delivery of useful software.
* Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
* Product is developed fast and frequently delivered (weeks rather than months.)
* A face-to-face conversation is the best form of communication.
* It continuously gave attention to technical excellence and good design.
* Daily and close cooperation between business people and developers.
* Regular adaptation to changing circumstances.
* Even late changes in requirements are welcomed.

**Disadvantages**

* There is a lack of intensity on necessary designing and documentation.
* It requires an expert project member to take crucial decisions in the meeting.
* Cost of Agile development methodology is slightly more as compared to other development methodology.
* The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.
* It is not useful for small development projects.

**Real-time Examples**

This model most prevalent in software industry to manage the workload and task assignment to particular teams and their members and this is done using various task manager which use this technology like Jira etc.

**V-MODEL**

**Introduction**

The V-model is a graphical representation of a systems development lifecycle. It is used to produce rigorous development lifecycle models and project management models. The V-model falls into three broad categories, the German V-Modell, a general testing model and the US government standard.

The V-model summarizes the main steps to be taken in conjunction with the corresponding deliverables within computerized system validation framework, or project life cycle development. It describes the activities to be performed and the results that have to be produced during product development.

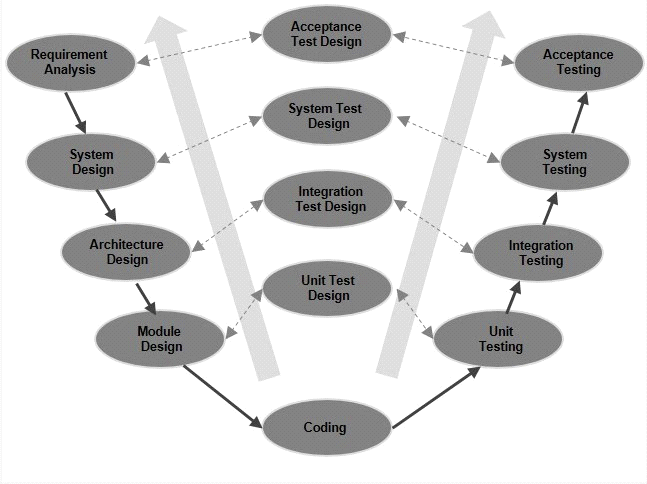
The left side of the "V" represents the decomposition of requirements, and creation of system specifications. The right side of the "V" represents integration of parts and their validation. However, requirements need to be validated first against the higher level requirements or user needs. Furthermore, there is also something as validation of system models

The V-model is a graphical representation of a systems development lifecycle. It is used to produce rigorous development lifecycle models and project management models. The V-model falls into three broad categories, the German V-Modell, a general testing model and the US government standard.

The V-model summarizes the main steps to be taken in conjunction with the corresponding deliverables within computerized system validation framework, or project life cycle development. It describes the activities to be performed and the results that have to be produced during product development.

The left side of the "V" represents the decomposition of requirements, and creation of system specifications. The right side of the "V" represents integration of parts and their validation. However, requirements need to be validated first against the higher level requirements or user needs. Furthermore, there is also something as validation of system models

**PHASE DIAGRAM**



**ADVANTAGES OF V-MODEL**

* Optimization of communication between participants through firmly-defined terms and responsibilities
* Minimization of risks and better plannability through firmly-prescribed roles, structures, and results
* Improvement of product quality through integrated quality assurance measures
* Cost savings through transparent reappraisal of the entire product life cycle
* Easy to Understand.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**DISADVANTAFGES OF V-MODEL**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.
* Not suitable for bigger and complex project
* The processes are institutionalized during the project and when the project is finished, they are abolished

**V-MODEL APPLICATIONS**

* The V-Model is mandatory for most tenders for public-sector software projects. As such, it’s an important tool for companies that develop software for agencies and ministries. It can be used for software projects of any size, whether in commerce, the military or the public sector.
* The model serves as a tool to facilitate the organization and execution of development, maintenance, and advancement of diverse IT systems.
* the V-Model can be applied to other development areas such as electronic or mechanical systems in research and science.

**Comparison between different type of SDLC Model**

| **Properties of Model** | **Waterfall Model** | **Incremental Model** | **Spiral Model** | **Agile Model** | **Rad Model** |
| --- | --- | --- | --- | --- | --- |
| Planning in early stage | Yes | Yes | Yes | Yes | No |
| Returning to an earlier phase | No | Yes | Yes | Yes | Yes |
| Handle Large-Project | Not Appropriate | Not Appropriate | Appropriate | Not Appropriate | Not Appropriate |
| Detailed Documentation | Necessary | Yes but not much | Yes | Yes | Limited |
| Cost | Low | Low | Expensive | Low | Low |
| Requirement Specifications | Beginning | Beginning | Beginning | Beginning | Time boxed release |
| Flexibility to change | Difficult | Easy | Easy | Easy | Easy |
| User Involvement | Only at beginning | Intermediate | High | Flexible | Only at the beginning |
| Maintenance | Least | Promotes Maintainability | Typical | Minimum | Easily Maintained |
| Duration | Long | Very long | Long | Max 1 month | Short |
| Risk Involvement | High | Low | Medium to high risk | Moderate | Low |
| Framework Type | Linear | Linear + Iterative | Linear + Iterative | Increment +  Iterative | Linear |
| Testing | After completion of coding phase | After every iteration | At the end of the engineering phase | Frequent | After completion of coding |