# SDLC

# C-TALK

Version 1.1

**Author:**
Shikhar Dhing(201452021)
Sunny Sankhlecha(201452061)

Indian Institute of Information Technology Vadodara

**Team members :**

| TEAM MEMBER | ID |
|---|---|
| Bhoopendra Singh | 201452020 |
| Shikhar Dhing | 201452021 |
| Venkata Sandeep | 201452037 |
| Anjali Kumari | 201452042 |
| Vipin Sahu | 201452051 |
| Prahlad | 201452052 |
| Sachin Jangid | 201452060 |
| Sunny Sankhlecha | 201452061 |
| Kenneth Tenny | 201452066 |

**Revision History**

| Version | Description | Date | Authors | Reviewers |
|---|---|---|---|---|
| 1.1 | SDLC | Shikhar Dhing (201452021) Sunny Sankhlecha (201452061) | ANJALI KUMARI , KENNETH TENNY | |

# Contents

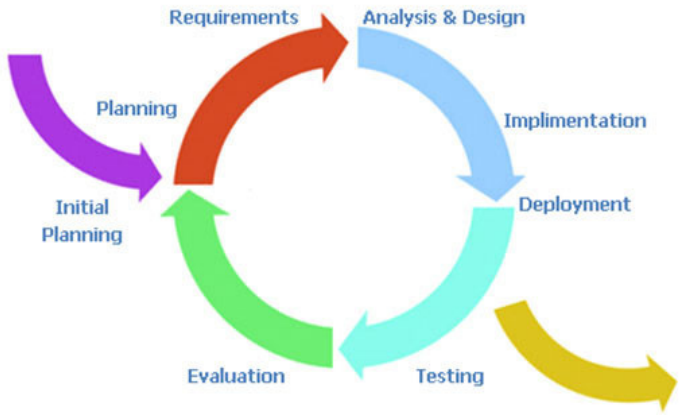
# 1 Software development life cycle

The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

# 2 Accepted model

## 2.1 Iterative waterfall model

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version




of the software for each cycle of the model.

## 2.2 Reason for acceptance

- For our project idea, we are not completely sure about all the requirements and specification. So we need a model where making modifications is easy and in Iterative waterfall model, it is easier to make corrections as the iterations progress. Our team can start working with the knowledge of initial requirement and by improving product step by step, we can track the defects at early stages. This avoids the downward flow of defects. Difficulty in design, coding and testing a

modification should signal the need for re-design or re-coding.

- Some working functionality can be developed quickly and early in the life cycle. Results are obtained early and periodically.We can plan Parallel development and progress can be measured.

- Less costly to change the scope/requirements. Testing and debugging during smaller iterations is easy. We can identify risks and resolve during iteration; and each iteration is an easily managed milestone. It is easier to manage risks. High risk part is done first where every increment operational product is delivered. Issues, challenges & risks identified from each increment can be utilized/applied to the next increment.

- We can analyse risks in a better way. It supports changing requirements. Initial Operating time is less. Better suited for large and mission critical projects.

- During life cycle software is produced early which facilitates customer evaluation and feedback. We can get the reliable user feedback. While presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.

# 3 Rejected Model

## 3.1 Waterfall Model

In a waterfall model, each phase must be completed fully before the next phase can begin. This type of model is basically used for the project which is small and there are no uncertain requirements.In this model the testing starts only after the development is complete. In waterfall model phases do not overlap

### 3.1.1 Reason for rejection

- In waterfall model once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought

out in the concept stage and we are not clear about our requirements so there is very much possibility of this situation.

- We cant produce working software until late during the life cycle and there is high amount of risk and uncertainty.
- It is not suitable for the projects just like where requirements are at a moderate to risk of changing. Poor model for long and ongoing projects because long project requires many changes in project with time.

## 3.2 Prototyping model

Basic idea of prototype is built to understand the requirements.prototype can enable the client to better understand the requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

### 3.2.1 Reason for rejection

- Because this type of model is used where there is no manual process or existing system to help determining the requirements and this is not a case in our project. So we can not use it.
- This model leads to implementing and then repairing way of building systems. It may increase the complexity of the system as scope of the system may expand beyond original plans
- Theres no ambiguity in our project definition and this kind of model is used for incomplete or inadequate problems. So we are cannot use this kind of model.

## 3.3 Rapid Application Development (RAD) Model

In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.

### 3.3.1 Reason for rejection

- Depends on individual performances for identifying project requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high

## 3.4 Evolutionary(Spiral) Model

The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations . The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral.

### 3.4.1 Reason for rejection

- It is a complicated approach especially for projects with a clear SRS. Which is not our case.
- Skills required, to evaluate and review project from time to time, need expertise.
- Rules and protocols should be followed properly to effectively implement this model. Doing so, through-out the span of project is tough.
- Due to various customizations allowed from the client, using the same prototype in other projects, in future, it may become difficult.
- It is not suitable for low risk projects.
- Meeting budgetary and scheduling requirements is tough if this development process is followed.

- Amount of documentation required in intermediate stages makes management of project very complex affair.