Abstract — This paper presents an overview information about existing processes, and life-cycle models, that support or could support software development. The software 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/product. Within the product lifecycle, software will undergo maintenance to correct errors and to comply with changes to requirements. The simplest overall form is where the product is just software, but it can become much more complicated, with multiple software developments each forming part of an overall system to comprise a Product. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

Keywords— Project Management, RAD model, SDLC, Spiral Model, Waterfall model, Agile Model, Application

I. INTRODUCTION

There are many development life cycle models that have been developed in order to achieve different requirement of objectives. The models specify the various stages of the process and the order in which they are carried out. The selection of model has very high impact on the testing that is carried out.

II. SDLC MODELS

SDLC selection is an approach or method or both by which software process model efficiently selected depends upon the given requirements and give better result rather than a normal selection process. The requirements consist of questions related to the thing that have been requested by the user for the project.

A. Waterfall Model

Waterfall approach was first Process Model in Software Engineering. In "The Waterfall" approach, the whole process of software development is divided into separate phases. The Phases in Waterfall model are: Requirement Specification Phase, Software Design, Implementation and Testing & Maintenance. All these phases are cascaded to each other so that second phase is started as and when defined set of goals achieved for first phase and it is signed off, hence we call it as “Waterfall Model”. The stages of "The Waterfall Model" are:

- Requirement Analysis.
- System & Software Design.
- Implementation & Unit Testing.
- Integration & System Testing.
- Operations & Maintenance.

I) Requirement Analysis: All possible requirements of the system to be developed are captured in this phase. Requirements are set of functionalities and constraints that the end-user expects from the system. The requirements are gathered from the end-user by consultation, these requirements are analysed for their validity and the possibility of incorporating the requirements in the system to be development is also studied. Finally, a Requirement
Specification document is created which serves the purpose of guideline for the next phase of the model.

2) System & Software Design: The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

3). Implementation & Unit Testing: On receiving system design documents, the work is divided in modules/units and actual coding is started. The system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality; this is referred to as Unit Testing. Unit testing mainly verifies if the modules/units meet their specifications.

4). Integration & System Testing: As above, the system is first divided in units which are developed and tested for their functionalities. These units are integrated into a complete system during Integration phase and tested to check if all modules/units coordinate between each other and the system works as per the specifications. After successfully testing the software, it is delivered to the customer.

5). Operations & Maintenance: Generally, problems with the system developed (which are not found during the development life cycle) come up after its practical use starts, so the issues related to the system are solved after deployment of the system. Not all the problems come in picture directly but they arise time to time and needs to be solved; hence this process is referred as Maintenance.

Advantages and Disadvantages

Advantages :

• Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
• Phases are processed and completed one at a time.
• Works well for smaller projects where requirements are very well understood.

Disadvantages :

• Once an application is in the testing stage, it is very difficult to go back and change something that was not well-through out in the concept stage.
• No working software is produced until late during the life cycle.
• High amounts of risk and uncertainty.
• Not a good model for complex and object-oriented projects.
• Poor model for long and ongoing projects.
• Not suitable for the projects where requirements are at a moderate to high risk of changing.

B. ITERATIVE MODEL-

An iterative lifecycle 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. Consider an iterative lifecycle model which consists of repeating the following four phases in sequence:

1). Requirements phase: In which the requirements for the software are gathered and analysed.

2). Design phase: Iteration should eventually result in a requirements phase that produces a complete and final specification of requirements is called design phase, in which a software solution to meet the requirements is designed. This may be a new design, or an extension of an earlier design.

3). Implementation and Test phase: Here the software is coded, integrated and tested.

4). Review phase: In which the software is evaluated, the current requirements are reviewed, and changes & additions are made to requirements proposed.
For each cycle of the model, a decision has to be made as to whether the software produced by the cycle will be discarded, or kept as a starting point for the next cycle. Eventually a point will be reached where the requirements are complete and the software can be delivered, or it becomes impossible to enhance the software as required, and a fresh start has to be made.

The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification of each version of the software against those requirements within each cycle of the model. Each cycle of the model produces software that requires testing at the unit level, for software integration, for system integration and for acceptance. As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of the software.

Advantages of Iterative model:

- In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
- In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
- In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- In iterative model less time is spent on documenting and more time is given for designing.

Disadvantages of Iterative model:

- Each phase of an iteration is rigid with no overlaps.
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle.

C. V-MODEL-

The V-model is a software development model which can be presumed to be the extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
Verification Phases-

1). Requirements analysis:

In this phase, the requirements of the proposed system are collected by analyzing the needs of the user(s). This phase is concerned about establishing what the ideal system has to perform. However, it does not determine how the software will be designed or built. Often, the users are interviewed and a document called the user requirements document is generated. The user requirements document will typically describe the system’s functional, physical, interface, performance, data, security requirements etc as expected by the user. It is one which the business analysts use to communicate their understanding of the system back to the users. The users carefully review this document as this document would serve as the guideline for the system designers in the system design phase. The user acceptance tests are designed in this phase.

2). System Design:

System engineers analyse and understand the business of the proposed system by studying the user requirements document. They figure out possibilities and techniques by which the user requirements can be implemented. If any of the requirements are not feasible, the user is informed of the issue. A resolution is found and the user requirement document is edited accordingly. The software specification document which serves as a blueprint for the development phase is generated. This document contains the general system organization, menu structures, data structures etc. It may also hold example business scenarios, sample windows, reports for the better understanding. Other technical documentation like entity diagrams, data dictionary will also be produced in this phase. The documents for system testing is prepared in this phase.

3). Architecture Design:

This phase can also be called as high-level design. The baseline in selecting the architecture is that it should realize all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology details etc. The integration testing design is carried out in this phase.

4). Module Design:

This phase can also be called as low-level design. The designed system is broken up in to smaller units or modules and each of them is explained so that the programmer can start coding directly. The low level design document or program specifications will contain a detailed functional logic of the module, in pseudo code - database tables, with all elements, including their type and size - all interface details with complete API references- all dependency issues- error message listings- complete input and outputs for a module. The unit test design is developed in this stage.

D. SPIRAL MODEL-

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral.

Fig 4. Spiral Model

1). Requirements: In this phase requirements are gathered.

2). Risk analysis Phase: In this phase a process is undertaken to identify risk and alternate solutions.

3). Engineering Phase: In this phase a prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase.

4). Evaluation phase: It allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Advantages of Spiral model:
- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
Additional functionality can be added at a later date.

Software is produced early in the software life cycle.

Disadvantages of Spiral model:
- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project’s success is highly dependent on the risk analysis phase.
- Doesn’t work well for smaller projects.

E. BIG BANG MODEL-

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

Advantages of Big Bang model:
- Big Bang testing has the advantage that everything is finished before integration testing starts.

Disadvantages of Big Bang model:
- The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

F. RAPID APPLICATION DEVELOPMENT (RAD) MODEL-

RAD is a linear sequential software development process model that emphasis an extremely short development cycle using a component based construction approach. It is a concept that products can be developed faster and of higher quality.

The traditional software development cycle follows a rigid sequence of steps with a formal sign-off at the completion of each. A complete, detailed requirements analysis is done that attempts to capture the system requirements in a Requirements Specification. Users are forced to "sign-off" on the specification before development proceeds to the next step. This is followed by a complete system design and then development and testing.

The phases in the rapid application development (RAD) model are:

1). Business modeling: The information flow is identified between various business functions.

2). Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.

3). Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

4). Application generation: Automated tools are used to convert process models into code and the actual system.

5). Testing and turnover: Test new components and all the interfaces.

RAD is a methodology for compressing the analysis, design, build, and test phases into a series of short, iterative development cycles. This has a number of distinct advantages over the traditional sequential development model. RAD projects are typically staffed with small integrated teams comprised of developers, end users, and IT technical resources. Small teams, combined with short, iterative development cycles optimizes speed, unity of vision and purpose, effective informal communication and simple project management.
Advantages of the RAD model:

- 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 of RAD model:

- Depends on strong team and individual performances for identifying business 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.

G. AGILE MODEL-

Agile development methodology attempts to provide many opportunities to assess the direction of a project throughout the development life cycle. Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks. Each iteration involves a cross functional team working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release at the end of each iteration. Multiple iterations might be required to release a product or new features.

Advantages of Agile model:

- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

H. PROTOTYPE MODEL-

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throw away prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an “actual feel” of the system, since the interactions with 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. The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.
Fig 7. Prototype Model

**Advantages of Prototype model:**

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified
- Requirements validation, Quick implementation of, incomplete, but functional, application.

**Disadvantages of Prototype model:**

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed
- Incomplete or inadequate problem analysis.

**XI CONCLUSIONS**

This was about the various SDLC models available and the scenarios in which these SDLC models are used. The information in this paper will help the project managers decide what SDLC model would be suitable for their project and it would also help the developers and testers understand basics of the development model being used for their project. This paper gives you an insight into the advantages and disadvantages of the SDLC models discussed.

Waterfall and V model are traditional SDLC models and are of sequential type. Sequential means that the next phase can start only after the completion of first phase. Such models are suitable for projects with very clear product requirements and where the requirements will not change dynamically during the course of project completion.

Iterative and Spiral models are more accommodative in terms of change and are suitable for projects where the requirements are not so well defined, or the market requirements change quite frequently.

Big Bang model is a random approach to Software development and is suitable for small or academic projects.

Agile is the most popular model used in the industry. Agile introduces the concept of fast delivery to customers using prototype approach. Agile divides the project into small iterations with specific deliverable features. Customer interaction is the backbone of Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment.

RAD (Rapid Application Development) and Prototype are modern techniques to understand the requirements in a better way early in the project cycle. These techniques work on the concept of providing a working model to the customer and stockholders to give the look and feel and collect the feedback. This feedback is used in an organized manner to improve the product.

**ACKNOWLEDGMENT**

I like to thank all authors who helped me to write this paper.

**REFERENCES**

[1]. Mr Joshua Boyde ,” A Down-To-Earth Guide To SDLC Project Management: Getting your system / software development life cycle project successfully across the line using PMBOK in an agile way”


[3]. Dorothy Gra, Ham,Erik Van Veenendaal, Isabel Evans, Rex Black “Foundations Of Software Testing (ISTQB Certification)”


[6]. Software Testing by Yogesh Singh, Cambridge University Press