A Comparative Study of Software Development Life Cycle Models

Jimeet Shah
Department of Computer Science and Engineering
Institute of Technology, Nirma University
17BCE033@nirmauni.ac.in

Abstract—A Software Development Life-Cycle (SDLC), also known as Software Process is an armature imposed upon the development of a software product [1]. The deciding factor for the survival of a specific “Software Development Model” in the Software Organisation as well as in the market is the Software Process. It may not be necessary that the set of steps/activities or in simple words, processes utilized by a software organization that has proven to be efficient and effective, would be applicable for another software organization. Other software organizations may opt for another set of processes, proving to be more convenient to them.

Software Process Model is used, to provide the basic structure, resulting in the creation of a plan from one stage to another. The significance of this plan is to provide a directive for planning, organizing and finally, executing the software product successfully.

In this paper, we have carried out a comparative analysis of the various traditional as well as modern software development models available like Waterfall, V-shaped, Iterative, Incremental, Evolutionary Prototyping, Spiral, Rapid Application Development (RAD) and also Agile models. As per SDLC, the performance of every mentioned model is calculated, based on their advantages and the disadvantages are taken into consideration.

Index Terms—Software Development Life Cycle, waterfall, v-shaped, iterative, incremental, prototyping, spiral, RAD, agile

I. INTRODUCTION

In this forthcoming era, the newly emerging information technologies have led to a sudden surge in the usage of the software. The impact of such development has resulted in software being an essential division of the Modern Human Civilization. In simple words, an entity depends immensely on the software due to the development of Information Technology.

Some of the major objectives of creating a software include: increasing the efficiency as well as the accuracy of the complex process that we are addressing. Also, the easier and faster execution of each step present in the life cycle of the respective software is a must. Here, we are introduced to the term "Software Development" [2].

Huge projects involving Software Development as one of the key ingredients, require an enormous amount of time and money to be invested, that had to be utilized in such a manner, that the final product is deliverable with the least-cost possible. With a no. of different approaches for presenting the software product, the main aim remains to be of abating cost and time, without compromising the quality of the finished product [3]. By taking the worst-case/crisis possible where funding is limited, even a little mishandling or poor selection of software models can lead to more exploitation of resources and time, which won’t be favorable.

A Software Development Life Cycle can be defined as a simplified structure laid down for developing a software product. In order to select, implement and in later stages, monitor the life cycle of a software i.e. based on the method chosen; an ISO standard "ISO 12207" is used for description [4]. For a given Software Development Process, the activities can be categorized as follows [1]:

1) Understanding the case
2) Decide a scheme for the solution
3) Coding for the designed solution
4) Testing the definite program

Since the above-mentioned activities are found to be much more complex in nature, they can be broken down into further sub-activities too. These sub-steps are created to increase the functionality and effectiveness of the software product. The basic activities can be written as:

1) Requirement analysis
2) Design
3) Coding
4) Testing
5) Maintenance

II. WATERFALL MODEL

The waterfall model was the first SDLC model to be introduced in Software Engineering. The first description of the model was done in 1970 in an article by Winston W.
Royce [5]. Although the term “waterfall” was never referred in it. The first use of the “Waterfall” was possibly done by Bell and Thayer [6].

The model is commonly referred to as a linear sequential model. It breaks the project into linear phases. Each phase depends on the output of the preceding phase. The outcome of each stage is known as deliverable. The phases are independent and do not overlap. Therefore the next phase cannot begin before the previous phase ends. In this way, the process flows in one direction, like a waterfall, hence the name “waterfall”.

1) Requirement Analysis:
The primary purpose of this phase is to comprehend the precise requirements of the client/customer and record them in a document that describes the system elaborately. The output of this phase is a document known as a Software Requirement Specification (SRS) document.

2) System Design:
The objective of this phase is to convert the specified requirements into an architecture that can be implemented. The complete architecture is designed along with the hardware and system requirements or specifications.

3) Implementation:
After the system design, the system is developed into small individual units. Each of these units is designed and tested independently.

4) Testing:
After the units of the system are developed, they are integrated into one unit which is tested for bugs and vulnerabilities. This phase is very essential as this determines whether the software will be released into the market or needs to be debugged.

5) Deployment:
Once the testing is done, the product is released into the market or deployed at the client end.

6) Maintenance:
This phase involves alteration of system or a component of the system to remove bugs generated during live usage, to improve performance of the system, or due to change requests by the customer.

Advantages
1) Simple and easy to understand
2) Works very well for smaller projects
3) Well defined stages
4) Heavily documented
5) Milestones are easily identifiable

Disadvantages
1) Real-world projects are seldom sequential
2) Working software produced only in the later stage of the life cycle
3) New requirements cannot be accommodated at a later stage
4) Turn out to be a poor model for long, complex and object-oriented projects

III. V-shaped Model

The V-Shaped Model is annexation made to the classical Waterfall Model. It is coined as the “Verification and Validation Model”. In comparison to the classical Waterfall model, the structure of this model isn’t linear, i.e. the process is worked out in the form of a branch. As shown in figure 2, it is quite visible that, the process steps are inclining upwards, after the coding stage, to form the V-Shape.

Within this model, the relationship between the developmental life cycle phase and its testing phase in terms of “Test Plan” is found. By moving in the horizontal direction (left to right), the progress with respect to time is observed; while in a vertical direction, the level of abstraction could be obtained.

1) Requirements:
Business requirements, as well as system requirements, are worked upon similarly as that in the Waterfall Model. Also, in this model, before getting started with the development phase, a “system test” plan is created which is focused solely on achieving the functionality as mentioned in the requirements.

2) High-Level Designing:
Being focused on the system’s structure and design helps us in providing the outline of the service, system, platform, product, and solution. Also, to examine the components of the software systems and their ability to work together, an Integrated Test Plan is created.

3) Low-Level Designing:
It is the phase where the definition of literal logic for each portion of the system exists. Also, the design of the actual components of software is done. For example, in this phase, all the possible relations between the classes and the methods represented through Class Diagrams are present. As a result, the Component testing plan is created.
4) Implementation:
This phase deals with all the coding parts. Once, the process of coding is completed, the execution path moves toward the right side of the V (in an upward direction) where all the test plans are present. Since these test plans were developed earlier, they are brought into effect now.

5) Coding:
It is the bottom-most part of the model. Here, the code for the given Module Design is developed by the developers. Later on, Unit testing is performed on the code written by developer [7].

Advantages
1) Higher chances of the model being successful compared to the Waterfall model because test plans are developed in the early stages of the life cycle
2) Feasible and straightforward in terms of usage
3) Each phase will always have a result
4) Suitable for small scale projects
5) Maintenance is optimum
6) Resource and capital control is higher

Disadvantages
1) Incorporation of changes after certain phases becomes difficult
2) Development is done during the implementation phase, i.e. production of a prototype isn’t possible in the early stages
3) No provision of a clear path for the problems found during test phases
4) The model is very rigid and little bit flexible

IV. INCREMENTAL MODEL

The Incremental model is a software development model where the software is split up into modules and the product is designed, implemented and tested incrementally till the complete product is ready. The product is divided into modules and each module goes through all phases of software development, namely, requirements, design, implementation, and testing.

The requirements of the product are broken down into components that can be developed incrementally. The planning is done for only the next increment and no long term plans are made. This makes it easier to modify the product as per needs. The first task that is undertaken is the development of core features. The product with basic core features is released. Then new functions are added into the next increments and versions of products are released. Each version has more additional features than the preceding function. The process is carried out until the final product is delivered to the client.

Advantages
1) Working product will be ready quickly
2) Flexible in nature
3) Easy testing and debugging
4) Risk management is simpler
5) Less costly if requirements change during production

Disadvantages
1) Requires very good planning
2) Requirements need to be well defined
3) Total cost is high
4) Time consuming

V. BIG BANG MODEL

The Big Bang model is a unique SDLC model which doesn’t follow any particular procedure. With inputs such as time, money and resources, the development phase gets started, with
the output either being a successful software product, as per the requirements stated by the client or not. Even one of the main reasons for using this model is when the requirements aren’t clear enough and the project team is very small. It is in such cases, that implementation takes place on the fly lacking analysis on all the necessary features. The need for planning is least as no specific protocol is followed. Loosely based on the theory and cosmological model going by the same name, the notion is: "Beginning with Null, a sudden rapid growth and development of code will swiftly emerge, hence producing a finished software product in an instant" [8].

Without much regard about the consequences that may arise, development of the product is done on a day-to-day basis. Every component of the product is prepared with a "Laissez-Faire" attitude (translated as "Let us do it"), with almost no regard for the future requirements and enjoying the work done at the moment. The model is suitable for products whose release date isn’t decided.

**Design and Application:**

The model comprises of focusing solely on utilizing resources over coding with the least time spent over planning. Depending upon the arrival of requirements and the level of understanding, the implementation will be carried out. Depending upon the degree of change to be made, revamping of the software product may or may not be done. Such a model is ideal for small scale projects where less number of team members are working together. Due to the absence of proper analysis, an imbalance in the developmental cycle of the product can be found [9].

1) In the developmental phase, the integration of modules is done because all the individual modules are developed entirely and aren’t integrated separately. 2) Testing of each module is done independently for defects or bugs. 3) If any error or defect is found, the module is disintegrated after finding the root cause of the problem.

**Advantages**

1) Simple model to implement  
2) Requires zero planning  
3) Huge flexibility for the developers  
4) Managing tasks is easier  
5) Requirement of resources is very less  
6) A good learning aid for newcomers or students  

**Disadvantages**

1) Uncertainty and the risk factor is very high  
2) Isn’t recommendable for complex, object-oriented projects  
3) Not suitable for large scale projects  
4) If requirements aren’t clear, may turn out to be expensive  
5) Proves to be meager in nature for long ongoing projects

**VI. Evolutionary Prototyping Model**

The Evolutionary Prototyping Model majorly focuses on building prototypes and getting feedback from the customers constantly throughout the project. Such feedback is carried out to make sure that, the functionality based on requirements, stated by the customer are present within the service/product.

It is based on this feedback, that the developers create a more polished product. The customer again, provides feedback for the same, thereby repeating the cycle. In other words, the prototype is observed to be "Evolving" towards the utmost system. The model consists of four stages which are stated as follows:

1) Recognition of the fundamental requirements:
   Although there might not be a possibility to list down all
the necessary details or requirements, being a continuous process, all the basic things must be identified, to make the project work.

2) Creating a prototype:
For ensuring the success of the project, several prototypes will be created with each one better than the one before.

3) Verifying a prototype:
With the help of experimentation and conducting some surveys, we can obtain the feedback. The people present in the target market would act as the users, thereby assisting whether the system is good or not.

4) Alterations in the prototype:
Based on the feedback received from the customers, the developer team will work at making additional iterations. These iterations will be required to construct a perfect prototype which was observed to be insufficient or incomplete in earlier iterations.

Advantages
1) Developments are visible to the customers after each iteration
2) The outcome of the project is satisfactory
3) Functionalities which are lacking can be identified and thus risk factor is decreased
4) Members of the team can communicate effectively
5) With prototyping and feedback going hand-in-hand, a better understanding of customer’s need is gained
6) Prototypes can be changed as well as discarded

Disadvantages
1) Customer may expect to get the product delivered sooner, because of the prototype developed in the early stages.
2) Based on the impression of the initial prototype, the client may lose interest in the final product
3) Slow and time taking process where the forecast of the completion date is unknown
4) It may stimulate excessive changing requests from the client
5) Due to requests of almost ever-changing requirements from the client, documentation may turn out to be poor in terms of quality
6) Integrity and security is weak
7) Couldn’t be incorporated in large-scale projects

VII. SPIRAL MODEL
The Spiral model was proposed by Barry Boehm in 1986 [10]. It is termed as an evolutionary software model that combines prototyping with a linear sequential model and supports risk handling. It inherits the iterative nature from prototyping and control and systematic aspects from the linear model. In the spiral model, the software is developed as incremental releases. It is represented diagrammatically as a spiral with loops. The number of loops varies with the project. At any point, the radius of the spiral denotes the cost of the project till now, and the angular dimension shows the progress made till now.

Each spiral represents one iteration and has the following phases:

1) Objective determination:
Each iteration or spiral starts with gathering the purpose and requirements for the cycle. The objectives are identified and studied, alternatives are explored and the constraints are identified.

2) Risk analysis:
After the evaluation of all possible solutions, the best one is selected. After this, the potential risks are identified and the mitigation strategies for these risks are planned and finalized.

3) Development and validation:
During this phase, the development of the product takes place depending upon the requirements gathered before. After development testing of the product is carried out.

4) Planning:
The developed product is evaluated by customer and feedback are recorded and analyzed. After this, planning for the next phases is carried out.

The spiral model is very effective when:
1) Project is large
2) Frequent product release is required
3) Risk evaluation is crucial
4) Requirements are complex
5) Changes may be required any time

Advantages
1) Better risk handling
2) Very effective for large projects
3) Flexible in requirements
4) Changes can be incorporated later
5) Involves customer feedback
6) Product is ready early

Disadvantages
1) Complex
2) Not suitable for smaller projects
3) High expertise required for risk analysis
4) Risk of overshooting the budget
5) Documentation is more due to many intermediate processes

VIII. RAPID APPLICATION DEVELOPMENT (RAD) MODEL

In the RAD model, the parts or the functions are materialized in parallel, in the form of mini-projects [11]. This system is a version of the "Incremental Model". A functioning prototype is built from the timely-boxed, delivered and gathered developments. With such quick building-up of components, the customers get to see, utilize and also provide feedback on their specifications and delivery.

Focusing on short duration development life-cycle, the model can achieve growth rapidly, due to the adaptation of steps from the classical waterfall model. The approach used in those steps is “Component based-Construct Approach”. If the requirements are made clear at the beginning, the process allows the developing team to accomplish the target in a short time, maybe even days [12]. The stages present in the model are as follows:

1) Communication:
The action which works on the business problem and the required information features that should be supported by the software [13].

2) Planning:
Being one of the most significant phases of the model, it deals with assigning different software teams to work on different functionalities/modules of the same project parallelly [11].

3) Modelling:
   - Business Modeling: -
     The flow of information is identified among various business components.
   - Data Modeling:
     Defining data objects based on the information received from the business model is done.
   - Process Modeling:
     To achieve the business-oriented objective, the data objects are converted, to attain the flow of business information. The definition is identified and then created for CRUD (stands for Create, Read, Update, Delete) of the data objects.

4) Generation of application:
The processed models received from the Processing stage, are converted into code and actual system by using the automated tools [13].

5) Testing and Turnover:
The examination of different test components and all their interfaces is looked after, to finalize the software product. Since each prototype is tested at every iteration, the overall time taken for testing is reduced [13].

Advantages
1) Cost-effective in nature
2) Resource and cost could be managed/controlled
3) Risk involvement is very low
4) Changes can be incorporated easily
5) No overlapping of phases is observed
6) Time frame is short
7) Manual coding is reduced due to the advanced automated tools available for generating and reusing of the codes
8) Each phase of its life cycle, delivery of higher priority function to the client is made possible
9) Defects are usually less in number due to prototyping in its nature.
10) Encourages provision of feedback by the client

Disadvantages
1) Requires highly skilled and trained developers
2) Due to poor documentation, tracking of progress and the problems accustomed to them is difficult
3) Will fail when the technical risk is high
4) Management complexity is very high
5) Client involvement during the entire life cycle of the project is a must
6) Not desirable for small-scale projects
7) Presence of time-box reduces the features delivered at each iteration, with the features being pushed to a later version to get finished
8) Highly dependent on modeling skills

IX. AGILE METHODOLOGY

Agile methodology is a practice of software development that encourages continuous iterations of development and testing throughout the life cycle of software development. Agile methodology promotes adaptive planning, evolutionary development, early delivery, and continuous improvement and involves a rapid response to change. Unlike the waterfall model, both development and testing activities take place
concurrently. The agile methodology is based on four core values as discussed in Agile Manifesto [14]:

1) Individual and team interactions over processes and tools
2) Working software over comprehensive documentation
3) Customer collaboration over contract negotiation
4) Responding to change over following a plan

The Manifesto for Agile Software Development is based on the following 12 principles [14]:

1) Customer satisfaction through early and continuous delivery of valuable software
2) Welcome changing requirements, even late in development
3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4) Business people and developers must work together daily throughout the project
5) Build projects around motivated individuals. Give them the environment and support they need, and trust them
6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7) Working software is the primary measure of progress
8) Agile processes promote sustainable development
9) Continuous attention to technical excellence and good design enhances agility
10) Simplicity - the art of maximizing the amount of work not done - is essential
11) The best architectures, requirements, and designs emerge from self-organizing teams
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Agile method involves an incremental and iterative process for software design. The process is broken down into individual models that are worked upon by designers. The customer frequently looks into the project and product and can make decisions and project changes. Documentation is given less priority as compared to software development. Every iteration has its testing phase and at the end of each iteration, a working product is shipped.

Advantages
1) Customer satisfaction as there is persistent release of useful software
2) Working software is released at smaller intervals
3) Interactions are given more priority than development
4) Greater involvement of stakeholders
5) Late changes and requirements are entertained
6) More flexibility
7) Continuous attention is given to quality and design

Disadvantages
1) Lack of necessary documentation
2) Requires expert personnel for making crucial decisions
3) Project can go off track if requirements and outcomes are not clear
4) It is more flexible, so cannot be easily incorporated in larger and traditional organizations
5) not useful for smaller projects
6) Cost may be more as compared to other models

Agile methodology has many models of development. Some of the extensively used ones are discussed.

A. Scrum
Scrum is an agile development model which aims at how tasks are managed in a team-based development environment. It consists of three main roles:
- Scrum Master: responsible for setting up the team, meetings and removing obstacles in the development process
- Product Owner: creates product backlogs i.e. requirements and prioritizes them
- Scrum Team: performs the designated task

The process of scrum:
1) Each iteration is known as sprint
2) Product backlog is created which contains a list of requirements
3) For each sprint, top requirements from product backlog is selected and turned into a sprint backlog
4) The team works according to the sprint backlog
5) Work check is done daily
6) At the end of sprint, a functional product is delivered

B. Extreme Programming (XP)
The Extreme Programming model is very effective when the requirements are constantly changing. It is very useful when the customer is not sure about the functional requirements of the system. Extreme programming involves frequent product releases increasing the productivity of the system and makes it possible to incorporate any customer requirement easily. The iterations are shorter (approx. 14 days) and each iteration has
6 phases. The requirements are gathered and known as stories and these stories are stored in what is known as a parking lot.

1) Planning:
   The stakeholders identified along with infrastructure requirements and agreements are made.

2) Analysis:
   Stories are gathered in the parking lot and are prioritized. Iteration time is fixed and resource planning is done.

3) Design:
   Tasks are identified, test scenarios are prepared.

4) Execution:
   The determined design is coded and unit testing is carried out.

5) Wrapping:
   Small releases are done, reviews are recorded and improvements are made based on reviews.

6) Closure:
   The product is launched along with production support.

---

**C. Dynamic Software Development Model (DSDM)**

Dynamic Software Development Model is a dynamic approach to develop systems through iterative rapid application development. The main emphasis of DSDM is continuous user involvement and interaction. This methodology believes that projects evolve best through collaboration between users and developers. The main aim of this model is to deliver projects within time and budget constraints with the changing requirements. This model is especially useful when the requirements cannot be fixed at the starting of system design. The DSDM has five phases:

1) Feasibility Study:
   It assesses the essential constraints and necessities of the design and then evaluates whether the application is a suitable candidate for DSDM.

2) Business Study:
   The business requirements are specified. After this, the basic architecture of the system is prepared. The maintainability level is also identified to set quality standards that will be followed throughout the process.

3) Functional Model Iteration:
   This phase involves building a prototype. The prototype is developed iteratively along with user reviews to get the desired system. The prototype is continuously improved through user feedback. This cycle is repeated until a satisfactory functional model is ready.

4) Design and Build Iteration:
   This phase involves engineering the prototype to suit the operating environment. The prototypes of functional modeling are further refined to achieve a satisfactory standard. The output of this phase a system that is ready for implementation.

5) Implementation:
   This is the final stage of development. In this stage, the system is deployed in the environment. If everything was delivered as per requirements, no further development takes place or else the process is repeated wherever required.

Often the DSDM is combined with extreme programming to get a hybrid model which combines the advantages of both DSDM and XP.

---

**D. Kanban**

Kanban is a work management methodology. It conceptualizes both the process that is being followed and the actual work going on through the process. The main objective of Kanban is to recognize and find constrictions in your process and solutions so that work can be finished with cost-effectiveness at optimal speed or throughput. Initially, Kanban was developed for the manufacturing industry by Taiichi Ohno, but it was later applied to IT industries and works in general. It was adapted in IT industry by David J. Anderson.

A vital requirement of kanban is real-time communication and complete transparency of work. The work items are
illustrated on a kanban board, allowing all team members to see the progress of every work at any time.

The Kanban methodology is based on four foundational principles.

1) Start with what you are doing now
2) Agree to pursue incremental, evolutionary change
3) Initially, respect current roles, responsibilities, and job titles
4) Encourage acts of leadership at all levels

The initial phase in the introduction of Kanban is to imagine the work process. This is done as a Kanban board comprising of whiteboard and stickers or cards. Each card on the board speaks of a task in a basic Kanban board model, there are three sections:

1) To Do: This segment records the tasks that are not yet begun (aka “backlog”)
2) Doing: Compromises of the task that are in progress
3) Done: Involves the tasks that are finished

The excellence of Kanban is that it very well may be applied to any procedure or methodology. Regardless of the current Agile strategy being utilized - Scrum, XP, and others, or other traditional techniques - Waterfall, iterative, and so forth - you can apply Kanban over that to steadily begin improving the procedures, diminish process duration and improve work.

X. CAPABILITY MATURITY MODEL (CMM)

The Capability Maturity Model is the benchmark set for measuring the maturity of the software process of an organization. The methodology is used, in order to develop and ameliorate an organization’s software development process. Based on some Key Processing Areas (KPA), an organization can be assessed upon the scale of five possible maturity levels. The scaled level of maturity indicates the maturity of a company towards its clients and the project they are assigned or working on. The organizations are ranked by each level with respect to its standardization of the processes, present within the subject area i.e. being assessed.

For the provision of guidance over certain aspects like gaining control of the processes, for developing as well as maintaining software, the CMM is used by organizations. Also, the model is responsible for assisting the organization in finding different ways of evolving towards the culture of Software Engineering and also achieving management excellence. By guiding the organizations in the selection of process improving strategies by determining the current process’s maturity and finding the issues which are hypercritical to the quality and improvement of the software.

By concentrating on a finite set of activities, working ambitiously to achieve them, an organization will observe a steady improvement in its organizational-wide software processes. Such improvement enables continuous and ever-lasting gains in the capability of a software process.

A. The Five Stages of Software Process Maturity

The CMM provides a framework for continuous process improvement i.e. based on many small evolutionary steps. These steps are further divided into five maturity levels, which defines a numeric scale for the measurement of the maturity and evaluating the capabilities of the software process.

The term "Maturity Level" refers to a well-defined evolutionary plateau, to achieve a mature software process. A set of goals is set at each maturity level, which when achieved, leads to the stabilization of an important frame/part of the software process. An organization’s process capability is increased, by establishing different components present in the software process by achieving a framework at each maturity level.

1) Level-1 (Initial):
The level doesn’t provide a stable environment for the development and maintenance of the software to the organization typically. Frequently difficult for the organization to provide an orderly engineering process i.e. making commitments which, results in a series of catastrophic outcomes. During such a crisis, projects return to testing and coding, rather than going as per planned procedures. With such issues, the success depends on how effective and seasoned the software team is and, also on an exceptional manager.
Sometimes, the capable software managers can withstand the pressure of taking shortcuts in the process; which when they leave the project, leads to destabilization. Even the strongly engineered-process isn’t able to overcome such instability which was caused by the presence of improper management practices. At this level, the capability isn’t a characteristic of the organization, but, the individuals working under the project.

2) Level-2 (Managed):
   Policies and procedures to be managed and implemented respectively for the software process are established at level-2. The process is categorized as an effective one if it has been measured, documented, trained, enforced, practiced, and its ability to improve. Project commitments are made, based on the outcomes of some previous projects and the requirements of the current one. A track of schedules, functionalities, and software costs of the project are monitored by the software manager. The organizations ensure that software project standards are followed and integrity of the client is maintained.
   The capability of a software process present in level-2 can be summed up as being disciplined as earlier successes can be repeated and stable planning and tracking of the project is found. Also, the current project’s process is under the surveillance of a project managing system, that follows realistic plans as per the performance in previous projects.

3) Level-3 (Defined):
   Documentation of the standard processes for development and maintenance of software across the organization is performed. The established processes at level-3 are utilized (and altered if required) to assist the technical-staff and managers effectively. Training programs are conducted, to ensure that the staff, as well as the managers, have the required set of skills and knowledge, to fulfill the roles assigned to them. While standardizing the software processes, the organization exploits effective practices of software engineering.
   A defined software process also called "Tailored Process" is developed, which includes a consistent integrated set of well-defined processes of Software Engineering. Such processes consist of standards and procedures for performing the task, readiness criteria, certain inputs, outputs and also mechanisms for verifying. Here, activities are stable and repeatable making the organizations listed in the level as standard and consistent. Based on the basic understanding of roles, responsibilities, and activities in the tailored process, the capability of the process is assessed.

4) Level-4 (Managed):
   The Managed Level deals with setting up the quantitative quality goals for processes as well as software products. As an important part of the organization’s measurement program, quality and the productivity of the software process activities across the projects are measured. At level-4, instrumentation of software processes with well-defined measurements is accomplished, establishing a quantitative foundation for evaluation purposes.
   The projects can achieve control over their products and processes by constricting the variations found in their process’ performance until they fall-within acceptable boundaries quantitatively. The risks present while moving up the learning curve of a fresh application domain are managed carefully. With the help of this level, the prediction of trends with respect to the processes as well as the product quality within the quantitative bounds is made possible. Software processes at this level are quantifiable and predictable due to the process being operated within the limits set. If known limits are exceeded, actions are taken to bring the situation under control. High-end quality products are found as an output.

5) Level-5 (Optimising)
   The main focus of an organization present at the level is continuous improvement of the process. To prevent any occurrences of defects, different means are used to identify the weaknesses, to fortify the process proactively [15]. The project team analyses the defects thereby evaluating the processes, to get to their causes. Also, the lessons learned and the defects found in the projects are disseminated and noted respectively for other projects. Innovations involving exploitation of the best software engineering practices are transferred throughout the system of organization.
   Aiming at the removal of chronic waste i.e. unacceptable, level-5 is focused on improvising the software process by eliminating "common variables" causing inefficiency. The capability of software at this level can be said to be continuously improving due to the striving efforts of level-5 organizations in improving the overall performance of their projects and process capabilities too [15]. The innovation in practices in the form of technology and the incremental advancements in current processes are responsible for improvements.
Advantages

1) Improvement in quality of software product
2) Highly flexible
3) Delivering of the product is on-time
4) Processes are improved in the organization
5) Productivity is increased
6) Overall returns are increased
7) Overlapping of phases isn’t possible

Disadvantages

1) Doesn’t reflect any information about the software
2) Recommended for repetitive tasks only
3) Can’t be utilized in emergencies i.e. time frame is very long
4) Focus is overreaching the next level, instead of improving the existing process
5) Doesn’t tell "how" to achieve the goals
6) Dependent on factors like training and documentation i.e. additional time required, if absent

XI. RATIONAL UNIFIED PROCESS (RUP)

Rational Unified Process is a software development process created by IBM in 2003. It is not a definite process, but a flexible, adjustable process framework. The project team can select the elements of the process as per their needs.

RUP is dependent on building blocks that describe what is to be produced, the requirements, and how certain development goals are to be attained. The main building blocks are:

1) Roles (who): roles define skills and responsibilities
2) Work products (what): the resultant of a task, including documents and models produced
3) Tasks (how): task defines work assigned to roles

The RUP consists of four phases. The phases are similar to the waterfall model but the essence of RUP lies in the iterations that take place within phases. The RUP phases are visualized as a RUP hump chart.

Advantages

1) Less time for integration as integration process goes on throughout the life cycle
2) Development time is less due to reuse of components
3) Allows to deal with changing requirements efficiently
4) Improved risk management
5) Emphasizes on accurate documentation

Disadvantages

1) Expert team members required
2) Complex development process
3) Component reuse will not be possible in projects which utilize cutting edge technology
4) Integration throughout the cycle sounds good but on bigger projects, it will lead to confusion
<table>
<thead>
<tr>
<th>Properties of Model</th>
<th>WaterFall Model</th>
<th>V-Shaped Model</th>
<th>Incremental Model</th>
<th>Big Bang Model</th>
<th>Spiral Model</th>
<th>Rad Model</th>
<th>Prototyping</th>
<th>CMM</th>
<th>RUP</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning in early stage</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Returning to an earlier phase</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Handle Large-Project</td>
<td>Not Recommendable</td>
<td>Not Recommendable</td>
<td>Not Recommendable</td>
<td>Not Recommendable</td>
<td>Not Recommendable</td>
<td>Not Recommendable</td>
<td>Recommend for higher levels</td>
<td>Can become complicated</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Detailed Documentation</td>
<td>Necessary</td>
<td>Yes but not much</td>
<td>No</td>
<td>Yes</td>
<td>Limited</td>
<td>Limited</td>
<td>Present in higher levels</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>Low</td>
<td>Expensive</td>
<td>Low</td>
<td>Low</td>
<td>Expensive</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Very high</td>
<td></td>
</tr>
<tr>
<td>Requirement Specifications</td>
<td>Beginning</td>
<td>Beginning</td>
<td>Beginning</td>
<td>May change frequently</td>
<td>Beginning</td>
<td>Time boxed release</td>
<td>Changes Frequently</td>
<td>At level-2</td>
<td>Beginning</td>
<td>Changes frequently</td>
</tr>
<tr>
<td>Flexibility to change</td>
<td>Difficult</td>
<td>Little flexible</td>
<td>Easy</td>
<td>High</td>
<td>Easy</td>
<td>Highly flexible</td>
<td>Highly flexible</td>
<td>Considerable</td>
<td>Highly flexible</td>
<td></td>
</tr>
<tr>
<td>User Involvement</td>
<td>Only at beginning</td>
<td>Only at beginning</td>
<td>Intermediate</td>
<td>High</td>
<td>Only at the beginning</td>
<td>High</td>
<td>Only at the beginning</td>
<td>High</td>
<td>At beginning and at last</td>
<td>High</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Least</td>
<td>Low</td>
<td>Promotes Maintainability</td>
<td>Typical</td>
<td>Easily Maintained</td>
<td>Routine maintenance</td>
<td>Typical</td>
<td>At last phase</td>
<td>Easily maintained</td>
<td></td>
</tr>
<tr>
<td>Risk Involvement</td>
<td>High</td>
<td>Low</td>
<td>Medium to high risk</td>
<td>Medium to high risk</td>
<td>Low</td>
<td>Low</td>
<td>Varies according to the level</td>
<td>In early stages</td>
<td>Less</td>
<td></td>
</tr>
<tr>
<td>Framework Type</td>
<td>Linear</td>
<td>Branch type</td>
<td>Linear and Iterative</td>
<td>Linear + Iterative</td>
<td>Linear</td>
<td>Linear and Iterative</td>
<td>No specific framework followed</td>
<td>Linear + Iterative</td>
<td>Incremental and Iterative</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>After completion of coding phase</td>
<td>After each phase testing plan is initiated</td>
<td>After every iteration</td>
<td>After completion of coding phase</td>
<td>At the end of the engineering phase</td>
<td>After completion of coding</td>
<td>After completion of development phase</td>
<td>Present in higher levels</td>
<td>In last phase</td>
<td>Along with development</td>
</tr>
<tr>
<td>Overlapping Phases</td>
<td>No</td>
<td>No</td>
<td>Yes (As parallel development is there)</td>
<td>No phases present</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Re-usability</td>
<td>Least possible</td>
<td>To some extent</td>
<td>To some extent</td>
<td>No</td>
<td>To some extent</td>
<td>Yes</td>
<td>Weak</td>
<td>Yes</td>
<td>Yes</td>
<td>Reusable use-case</td>
</tr>
<tr>
<td>Time-Frame</td>
<td>Very Long</td>
<td>According to project size</td>
<td>Long</td>
<td>Short</td>
<td>Long</td>
<td>Short</td>
<td>Short</td>
<td>Short</td>
<td>Quite long</td>
<td>Short</td>
</tr>
<tr>
<td>Objective</td>
<td>High Assurance</td>
<td>Project transparency, risk reduction with quality of software</td>
<td>Rapid Development</td>
<td>Rapid Development</td>
<td>High Assurance</td>
<td>Rapid Development</td>
<td>Provision of a product at the end of each test cycle</td>
<td>Varies according to the level</td>
<td>High quality software within budget and time frame</td>
<td>Rapid and frequent delivery of solutions</td>
</tr>
<tr>
<td>Expertise required</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Less</td>
<td>High</td>
<td>Medium</td>
<td>Varies according to level</td>
<td>High</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
XII. Conclusion

Traditional models like Waterfall and V-Shaped models are sequential in nature, they are brought into effect when the client requirements are very clear and won’t change during the course of project dynamically. Sequential means that the next phase could be initiated just after the completion of the phase before. RAD and Evolutionary prototyping models are example of modern techniques, which help the client to attain clarity over the requirements of the project. It involves creation and provision of a working model, at every iteration of the life cycle to the clients, for collecting feedback on the look and feel of the product, thereby improvising it with the feedback provided. For smaller projects with very small team available, Big Bang model can be used to develop the Software product.

For clients who aren’t having clarity with the requirement of the project or, where the changes in market are very frequent, Iterative and Spiral models will be very effective. Nowadays, the most popular approach used in the industry goes by the name “Agile”. By bringing prototype approach into effect, concept of delivering the product faster is introduced under the model. Dividing the project into finer iterations, features to be delivered at each iterations are assigned respectively. The backbone of the methodology being customer feedback, typical features like open communication along with minimum documentation in such developmental environment is taken care of.

Many SDLC models are available for creating different software frameworks, all depending upon the weight-age of factors involved in the process. For example, an organisation would prefer V-shaped model , in case it gives higher priority to verification and validation of the model. With each model, having its own set of advantage and disadvantages, there’s always a possibility that the model chosen by the organisation maybe a hybrid/fusion of two or more models. Also, timing plays a major role in the development cycle of a software. In case of a delay, it may be possible that the competitor make take over the market, resulting in a huge loss to the organisation. If the software product having a bug, is released into the market way before the competitor, it may cause harm to the reputation of the organisation [16].

So, there will always be a trade-off between the quality as well as development phase of the software product. As a matter of fact, the clients don’t hope for a ”bug-free” product but, a user-friendly product which results in elation of the customers.

References