CHAPTER – 3
SOFTWARE MODELS ANALYSIS
CHAPTER – 3 SOFTWARE MODELS ANALYSIS

3.1 INTRODUCTION

As the continuation of review of literature, the analysis of different software development models along with their advantages and issues were discussed in this chapter. Software development models are playing major role to the development and resource allocation to produce the high quality software based on the requirement analysis. Software development processes is integrated and inter related process of human skill set, domain knowledge, computing skill, analytical process and conversion of concepts into process functionality, development of code and optimization resources. The development process are widely adopted major few models namely water fall, spiral, V-Model and integrative model. The initial process of all the models is the analysis phase but the further development and execution of process approaches are differing from model to model, this chapter provides the details of each model and is related merits and demerits inline to the development sectors. The first part of the chapter describes the activities of the software development.

3.2 SOFTWARE DEVELOPMENT ACTIVITIES

Software development processes are involved between three levels in an organization. The clients are providing the requirement to the executives of development concern. The executives are transfer the requirement into a proposal for the internal members of the development organization and the feasibility study are made. As per the feasibility report the major activities are computed in the software development firm. The major tasks are planning a project, analysis and design, implementation at client site, testing and document the content.
Planning is an important responsibility to prepare and mapping of available resources to execute the task in creating a software program. It is extracting the requirements from the customer is a basic step for planning. After getting all the general requirements from the client planning process are started at the software development firm. An analysis of the scope of the development determined and clearly stated as part of scope document of the project. This document can be considered a legal document for future clarification for both client and developer. [IAN 2001].

Planning process needs a foresight of the executive aware the existing strength of software development firm, their service, available skilled developers and ongoing development process. The planning processes map the existing, estimated and expected resources and infrastructure. As part of the planning process, feasibility study made in all aspects and proposal are prepared for the development process.

The analysis process identifies the functional specification, domain specification, technical and technological requirement to develop and computed the committed software projects. The analysis process initially provides the understanding of the requirement and the possibilities of the development with the specified terms and condition of clients. The requirement analysis process made the detailed analysis of the development process which includes module specification, data base design, development of code, testing process till the deployment process. Implementation is the realization of an application or execution of plan by the software engineers with the coding of the project. Software testing is an integral and important phase of the software development process. This part of the process ensures that defects are recognized as soon as possible. Documenting is the purpose of maintaining the details of work done in internal design of software throughout the development. It will also
help for future maintenance and enhancement. Maintaining and enhancing software to cope with newly discovered faults or requirements can take significant time and effort. The missed requirements may force redesign of the software. The ways in which processes are developed and executed are considered development architecture. The development architecture covers the process of software development called as software development life cycle (SDLC). The software development life cycle process and its specifications are described further.

### 3.3 SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

Software development activities start from the mobilization of project from client, development and end with delivery process are called as software development life cycle. This process is applicable for all the development process but the method of development differing from one project to another because of the nature of the problem, domain, available skilled developers, technological limitations and requirements specifications. These software engineering approaches and development models are not classified or regularized for the specific domain or a problem. The development concern or the software practitioners are selecting or adopting a development model based on their development and software industry knowledge. The software development process, functionalities are unique from project to project through the domain and problems are sometimes same. The development models are specify the phases, activities, sub activities, development milestones, accuracy, method of verification and validation in the entire development process.[CHRI 2002].

The models describe the phases of the software life cycle and the order in which those phases are executed. There are plenty of models, and many companies accept their own, but all the models carry very similar activities. Each of these methodologies has its place in modern software development and the most appropriate development process should be
applied to each project. Since the quality, reliability and time are key drivers for a successful project and these life cycles are used to obtain them the selection of software model influences the quality of services.

3.3.1 General Life Cycle Model

The Life cycle model follows phase by phase activities. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

Fig: 3.3.1 General Life cycle Model

**Requirements:** Business requirements of the customers are gathered in this phase. This phase is the main focal point of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements. The requirements are the explanations of the following questions like, who is going to use the
system? How will they use the system? What data should be input into the system? What data should be output by the system? These answers produces a nice big list of functionality that the system should provide, which describes functions of the system, business logic that processes data, what data is stored and used by the system, and how the user interface should work. The overall result is the system as a whole and how it performs, not how it is actually going to do it.

**Design:** The software system design phase which is produced from the results of the requirements phase. The developer’s maximum work is during this phase and this is the phase in which gives shape for their ideas. This is where the details on how the system will work are produced. Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

**Implementation:** Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle. This is the phase which gives value for overall activities and work done before. It makes the system to have life with full of real activities.

**Testing:** This phase comes after implementation phase. During this phase of testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. Unit tests and system/acceptance tests are done during this phase. Unit tests act on a specific component of the system, while system tests act on the system as a whole.
3.4 TYPES OF SOFTWARE DEVELOPMENT LIFE CYCLES (SDLC)

Due to the different types of projects requirements; the developers required to choose the SDLC phases according to the specific needs of the project. These different requirements and needs give us various software development approaches to choose from during software implementation. There are several models for developing the software products. Here we can see some four software development life cycles models.

- Water-fall Model
- Spiral Model (SM)
- V-Shaped Model
- Iterative Model
- Rapid Application Development (RAD)

3.4.1 WATERFALL MODEL

The waterfall model is a sequential development process, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design (validation), Construction, Testing and maintenance. The following picture illustrates the concept.
In Royce's original waterfall model, the following phases are followed in order:

1. Requirements specification
2. Design
3. Construction (coding)
4. Integration
5. Testing and debugging
6. Installation
7. Maintenance

To follow the waterfall model, one proceeds from one phase to the next phase for delivering the project in a way to meet all of the requirements given. When the offer is fully
completed, an implementation of that design is made by the developers. Thus the waterfall model maintains that one should move to a phase only when it’s proceeding phase is completed and perfected. However, there are various modified waterfall models (including Royce’s final model) that may include slight or major variations upon this process.

**Advantages:**

a. Easy to explain to the user

b. Structures approach

c. Stages and activities are well defined

d. Helps to plan and schedule the project

e. Verification at each stage ensures early detection of errors / misunderstanding

**Disadvantages:**

a. Assumes that the requirements of a system can be frozen

b. Very difficult to go back to any stage after it finished.

c. Little flexibility and adjusting scope is difficult and expensive.

d. Costly and required more time

**3.4.2 SPIRAL MODEL**

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects. [BAR 1988].
The steps in the spiral model can be generalized as follows:

The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system. A preliminary design is created for the new system. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
A second prototype is evolved by a fourfold procedure:

a. evaluating the first prototype in terms of its strengths, weaknesses, and risks;

b. defining the requirements of the second prototype;

c. planning and designing the second prototype;

d. Constructing and testing the second prototype.

**Advantages:**

a. Estimates (i.e. budget, schedule, etc.) become more realistic as work progresses, because important issues are discovered earlier.

b. Early involvement of developers.

c. Manages risks and develops system into phases.

**Disadvantages:**

a. High cost and time to reach the final product.

b. Needs special skills to evaluate the risks and assumptions.

c. Highly customized limiting re-usability.

### 3.4.3 V-Model

It is an extension for 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 major difference between v-shaped model and waterfall model is the early test planning in v-shaped model. [GOU 2012].
Advantages:

a. Simple and easy to use.

b. Each phase has specific deliverables.

c. Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.

d. Works well for where requirements are easily understood.

Disadvantages:

a. Very inflexible, like the waterfall model.

b. Little flexibility and adjusting scope is difficult and expensive.
c. Software is developed during the implementation phase, so no early prototypes of the software are produced.

d. Model doesn’t provide a clear path for problems found during testing phases.

e. Costly and required more time, in addition to detailed plan

3.4.4 Iterative Model

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or versions of the system.[IAN 2001]

Fig: 3.4.4 Iterative Model Development Process
Iterative development slices the deliverable business value (system functionality) into iterations. In each iteration a slice of functionality is delivered through cross-discipline work, starting from the model/requirements through to the testing/deployment. The unified process groups iterations into phases: inception, elaboration, construction, and transition.

- **Inception** identifies project scope, risks, and requirements (functional and non-functional) at a high level but in enough detail that work can be estimated.
- **Elaboration** delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements.
- **Construction** incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
- **Transition** delivers the system into the production operating environment.

Each of the phases may be divided into 1 or more iterations, which are usually time-boxed rather than feature-boxed. Architects and analysts work one iteration ahead of developers and testers to keep their work-product backlog full.

**Advantages:**

a. Produces business value early in the development life cycle.

b. Better use of scarce resources through proper increment definition.

c. Can accommodate some change requests between increments.

d. More focused on customer value than the linear approaches.

e. Problems can be detected earlier
Disadvantages:

a. Requires heavy documentation. Follows a defined set of processes
b. Defines increments based on function and feature dependencies
c. Requires more customer involvement than the linear approaches
d. Partitioning the functions and features might be problematic
e. Integration between iteration can be an issue if this is not considered during the development.

3.5 RAPID APPLICATION DEVELOPMENT (RAD)

Rapid Application Development (RAD) is a software development methodology, which involves iterative development and the construction of prototypes. Rapid application development is a term originally used to describe a software development process introduced by James Martin in 1991.

Basic principles: Key objective is for fast development and delivery of a high quality system at a relatively low investment cost.

1 Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
2 Aims to produce high quality systems quickly, primarily through the use of iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. These tools may include Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools and object-oriented techniques.
3 Key emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.
4 Project control involves prioritizing development and defining delivery deadlines or “time boxes”. If the project starts to slip, emphasis is on reducing requirements to fit the time box, not in increasing the deadline.
5 Active user involvement is imperative.
6 Iteratively produces production software, as opposed to a throwaway prototype.
7 Produces documentation necessary to facilitate future development and maintenance.
8 Standard systems analysis and design techniques can be fitted into this framework.
### 3.6 COMPARISON OF VARIOUS SOFTWARE DESIGN MODELS

<table>
<thead>
<tr>
<th>Name of models</th>
<th>Requirement specification</th>
<th>Design</th>
<th>Construction</th>
<th>Integration &amp; testing</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Waterfall model</td>
<td>1. All possible requirements of the system to be developed are captured in this phase. 2. Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.</td>
<td>1. The requirement specifications from first phase are studied in this phase and system design is prepared. 2. The system design specifications is created to serve as input for the next phase of the model.</td>
<td>1. 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;</td>
<td>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 as a whole behaves as per the specifications.</td>
<td>This phase of &quot;The Waterfall Model&quot; is virtually never ending phase (Very long). 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.</td>
</tr>
<tr>
<td>Name of models</td>
<td>Requirement specification</td>
<td>Design</td>
<td>Construction</td>
<td>Integration &amp; testing</td>
<td>Maintenance</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------</td>
<td>--------</td>
<td>--------------</td>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>2. Spiral model</td>
<td>1. The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.</td>
<td>1. A preliminary design is created for the new system.</td>
<td>1. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product. 2. A second prototype is evolved by a fourfold procedure: (1) evaluating the first prototype in terms of its strengths, weaknesses, and risks; (2) defining the requirements of the second prototype; (3) planning and designing the second prototype; (4) constructing and testing the second prototype.</td>
<td>1. The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary, another prototype is developed from it according to the fourfold procedure outlined above. 2. The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired. 3. The final system is constructed, based on the refined prototype. 4. The final system is thoroughly evaluated and tested.</td>
<td>Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.</td>
</tr>
<tr>
<td>Name of models</td>
<td>Requirement specification</td>
<td>Design</td>
<td>Construction</td>
<td>Integration &amp; testing</td>
<td>Maintenance</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>3. Iterative model</td>
<td>The requirements for the software are gathered and analyzed. Iteration should eventually result in a requirements phase that produces a complete and final specification of requirements.</td>
<td>A software solution to meet the requirements is designed. This may be a new design, or an extension of an earlier design.</td>
<td>-------------------</td>
<td>when the software is coded, integrated and tested.</td>
<td>Review phase, in which the software is evaluated, the current requirements are reviewed, and changes and additions to requirements proposed.</td>
</tr>
<tr>
<td>4.V-Model</td>
<td>1. The requirements of the proposed system are collected by analyzing the needs of the user(s). 2. The user</td>
<td>1. System engineers analyze and understand the business of the proposed system by studying the user requirements document. 2. They figure out</td>
<td>1. The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly. 2. The unit test design is developed in this stage.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name of models</td>
<td>Requirement specification</td>
<td>Design</td>
<td>Construction</td>
<td>Integration &amp; testing</td>
<td>Maintenance</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>5.RAD model</td>
<td>acceptance tests are designed in this phase.</td>
<td>possibilities and techniques by which the user requirements can be implemented.</td>
<td>Data Modeling: The information collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified and the relation between these data objects (entities) is defined.</td>
<td>Process Modeling: The data object defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting or retrieving a data object.</td>
<td>Testing and Turn over: Many of the programming components have already been tested since RAD emphasis reuse. This reduces overall testing time. But new components must be tested and all interfaces must be fully exercised.</td>
</tr>
</tbody>
</table>
3.7 DISCUSSIONS ON SOFTWARE MODELS:

The analysis of the models provides the view on the software model usage and its limitations [NAB 2010]. The limitation of the costing and time factors are attempted to resolve with new systematic approaches on the development models. The traditional models such as water-fall model, spiral model, V-model and Iterative model has common approach of phase based development. However, the intermediate process are differ from one to another. To overcome the difference in the activities and the phase relationship common phase based approach is attempted to evaluate the developers performance as part of the research. The development phase internal and in between activities weight and the corresponding correlation, regressive analysis process will provide the views on dependency. But the dependency alone can’t determine the quality. The phase activity weight and the internal influence on the phase to another phase are playing vital role on quality factors. The quality measures are attempted to derive based on the analysis of exiting model weight and its dependency. The calculated dependency and independent level values are try to modify the different models according to the phases of each model. The few activities are same in models or the models are integrated one with another and named another models. These merging and uniqueness of the model activities are aimed to compute and evaluate the software model. The analysis process and the evaluation is described in chapter four.

3.8 SUMMARY

This summary illustrates a substantial analysis has been expended by way of analysis case studies, and published reports. This highlights the need to understand more about the relationship between the activities of each phase of all the models. The following chapter will
explain in detail about the construction of associative matrix and level of dependency between the activities of phases.