Chapter 3

Technology Adapted

3.1 Introduction

With involving of this software engineering project technology adaptation can be categorized and can be discussed under following three main headings;

1. Software process models,
2. System analysis and design methodology ,
3. Unified Modeling Language, and
4. Development environment.

In this chapter it will also be described how some of these techniques are used to solve the problems in this software project and why some of these techniques are used to solve the problems in this software project.

3.2 Software Process Models

In chapter 4 of Software Engineering seventh edition of Professor Ian Someville; software process model is defined as "a set of activities that leads to the production of a software product" and three software process models are identified as follows;

1. The waterfall model.
2. Evolutionary development model.
3. Component based software engineering model [4].

3.2.1 The Waterfall Model

In "The Waterfall" approach, the whole process of software development is divided into separate process phases. The phases in Waterfall model are: Requirement Specifications 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 are achieved for first phase and it is signed off, so the name "Waterfall Model".

All the methods and processes undertaken in Waterfall Model are more visible.

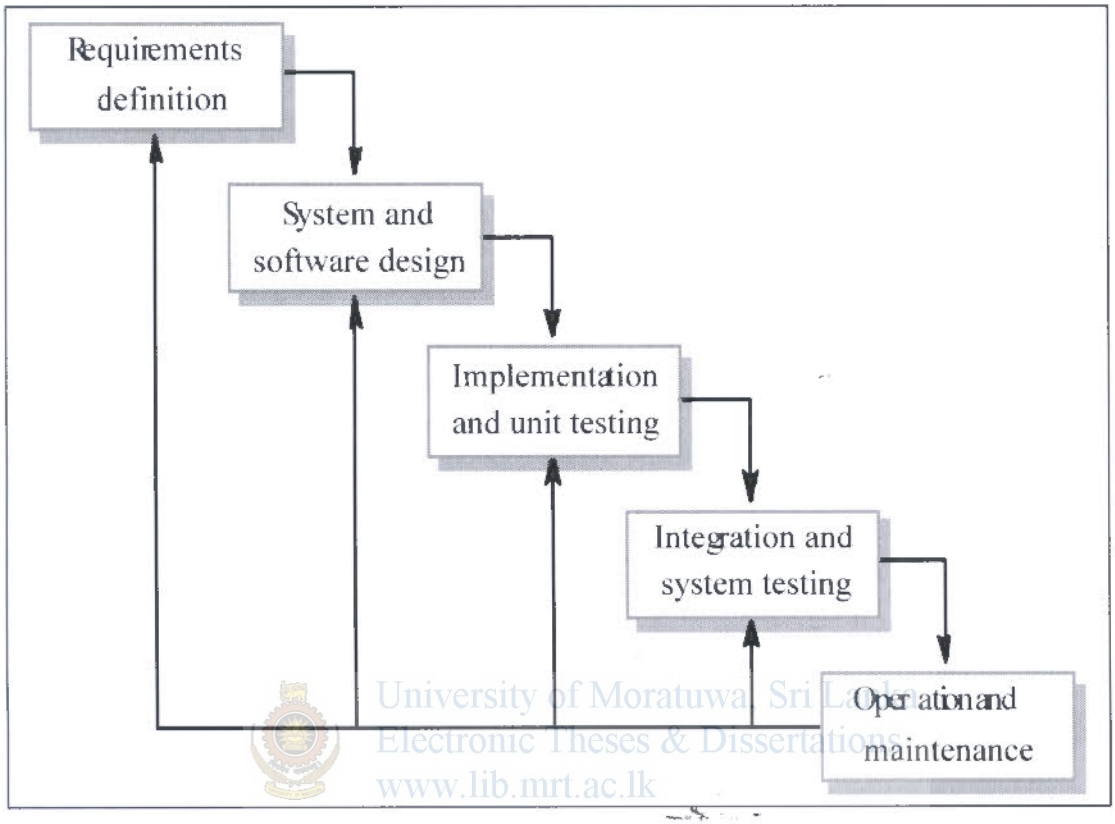

Figure 3.1 – Waterfall Model

The stages of "The Waterfall Model" are as follows [5]:

- **Requirement Analysis & Definition**

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 (who will be using the system) expects from the system. The requirements are gathered from the end-user by consultation, these requirements are analyzed 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.

- **System & Software Design**

Before a starting for actual coding, it is highly important to understand what we are going to create and what it should look like? 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.

- **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.

- **Integration & System Testing**

As specified 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 as a whole behaves as per the specifications. After successfully testing the software, it is delivered to the customer.

- **Operations & Maintenance**

This phase of "The Waterfall Model" 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. 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 of waterfall model**

Main advantage of using waterfall model is documentation is produced at each phase and to fit with other engineering models.

- **Disadvantages of the Waterfall Model**

1) As it is very important to gather all possible requirements during the Requirement Gathering and Analysis phase in order to properly design the system, not all requirements are received at once, the requirements from customer goes on getting added to the list even after the end of "Requirement Gathering and Analysis" phase, this affects the system development process and its success in negative aspects.

2) The problems with one phase are never solved completely during that phase and in fact many problems regarding a particular phase arise after the phase is signed off, these results in badly structured system as not all the problems (related to a phase) are solved during the same phase.

3) The project is not partitioned in phases in flexible way.

4) As the requirements of the customer goes on getting added to the list, not all the requirements are fulfilled, this results in development of almost unusable system. These requirements are then met in newer version of the system; this increases the cost of system development.

3.2.2 Evolutionary Development Model

Evolutionary development is based on the idea of developing an initial implementation, exposing this to user comments and refining it through many versions until an adequate system has been developed. Specification, development and validation activities are interleaved rather than separate with rapid feed back across activities.

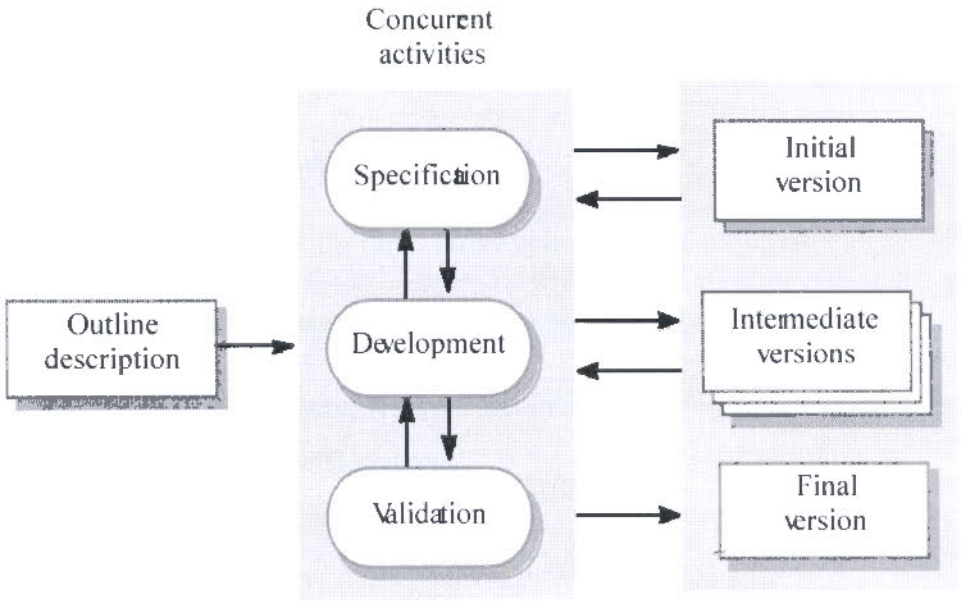

Figure 3.2 - Evolutionary Development Model

There are two fundamental types of evolutionary development have been introduced.

- **Exploratory Development**

The objective of the process is to work with the users explore their requirements and deliver a final system.

- **Throw-away Prototyping**

Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

- **Advantages**

For small or medium-size interactive systems.

For parts of large systems (e.g. the user interface).

For short-lifetime systems.

- **Disadvantages**

Lack of process visibility.

Systems are often poorly structured.

Special skills (e.g. in languages for rapid prototyping) may be required.

3.2.3 Component Based Software Engineering Model

Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.

Process stages.

Component analysis.

Requirements modification.

System design with reuse.

Development and integration.

This approach is becoming increasingly used as component standards have emerged.

Main advantages of the component based software engineering are reducing the software to be developed and there by reducing the cost and the risk

3.2.4 Comparison of Software Process Models

Following is a comparison of waterfall, evolutionary and component based software process modules.

Table 3.1 – Comparison of Software Process Models

Features	Waterfall model	Evolutionary development model	Component based software Engineering
Partitioning of the project	Can be done	Cannot be done	Cannot be done
System design with re-use	Can be done	Cannot be done	Can be done
Special skills	required	Not required	Not required
Process Visibility	High	low	Medium
Requirement modification	Cannot be done	Can be done	Can be done

3.3 System Analysis and Design Methodology

3.3.1 Object Oriented Analysis & Design

"Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects" [6].

Each object represents some entity of interest in the system being modeled, and is characterized by its class, its state (data elements), and its behavior. Various models can be created to show the static structure, dynamic behavior, and run-time deployment of these collaborating objects. There are a number of different notations for representing these models, such as the Unified Modeling Language (UML).

- **Object Oriented Analysis (OOA)**

Object-oriented analysis (OOA) looks at the problem domain, with the aim of producing a conceptual model of the information that exists in the area being analyzed [6]. At this stage, to understand the problem domain an analysis has been carried. The sources of information were taken by interviewing the current users, through group discussions and observations.

The final out come of the Object Oriented Analysis is a description of what the system functionality required to do in the form of a conceptual model. After OOA it was presented by using Use case diagrams and use case descriptions.

- **Object Oriented Design (OOD)**

"Object-oriented design (OOD) transforms the conceptual model produced in object-oriented analysis to take account of the constraints imposed by the chosen architecture and any non-functional – technological or environmental – constraints, such as transaction throughput, response time, run-time platform, development environment, or programming language" [6].

The concepts in the analysis model are mapped onto implementation classes and interfaces. The result is a model of the solution domain, a detailed description of how the system is to be built.

3.3.2 Structured System Analysis and Design Methodology (SSADM)

Structured Systems Analysis and Design Method (SSADM) is a systems approach to the analysis and design of information systems [6]. The 3 most important techniques that are used in SSADM are:

- **Logical Data Modeling**

This is the process of identifying, modeling and documenting the data requirements of the system being designed. The data are separated into entities (things about which a business needs to record information) and relationships (the associations between the entities).

- **Data Flow Modeling**

This is the process of identifying, modeling and documenting how data moves around an information system. Data Flow Modeling examines processes (activities that transform data from one form to another), data stores (the holding areas for data), external entities (what sends data into a system or receives data from a system), and data flows (routes by which data can flow).

- **Entity Behavior Modeling**

This is the process of identifying, modeling and documenting the events that affect each entity and the sequence in which these events occur.

3.3.3 Comparison of OOAD and SSADM.

Table 3.2 – Comparison of OOAD and SSADM

Main Features	OOAD	SSADM
Main motivation	Object driven	Data driven
Dependability	Objects are independent	Data are interdependent
Usability	Objects can be reused	Data can be reused
Understandability	High	Medium
Maintainability	High	Medium

3.4 Unified Modeling Language (UML)

"Modeling is the designing of software applications before coding. Unified Modeling language helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements" [7]. It is very important to distinguish between the UML model and the set of diagrams of a system. A diagram is a partial graphical representation of a system's model. The model

also contains a "semantic backplane" — documentation such as written use cases that drive the model elements and diagrams. UML diagrams represent three different views of a system model;

- **Functional requirements view**

Represents the functional requirements of the system from the user's point of view.

Ex. use case diagrams (See Appendix 2 for - use case diagram)

- **Static structural view**

Emphasizes the static structure of the system using objects, attributes, operations and relationships.

Ex. class diagram (See Appendix 2 - class diagram)

- **Dynamic behavior view**

Emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.

Ex. sequence diagrams, activity diagrams diagram (See Appendix 2 - class diagram)

3.5 Development Environment

3.5.1 LAMP

The acronym LAMP refers to a solution stack of software, usually free and open source software, used to run dynamic Web sites or servers. The original expansion is as follows:

Linux, referring to the operating system.

Apache, the Web server.

MySQL, the database management system (or database server).

PHP or others, i.e. Perl, Python, the programming languages.

The combination of these technologies is used primarily to define a web server infrastructure, define a programming paradigm of developing software, and establish a software distribution package [8].

3.5.2 WAMP

WAMPs are packages of independently-created programs installed on computers that use a Microsoft Windows operating system. The interaction of these programs enables dynamic web pages to be served over a computer network, such as the internet or a private network.

"WAMP" is an acronym formed from the initials of the operating system (Windows) and the package's principal components: Apache, MySQL and PHP (or Perl or Python). Apache is a web server, which allows people with web browsers like Internet Explorer or Firefox to connect to a computer and see information there as web pages. MySQL is a database manager (that is, it keeps track of data in a highly organized way). PHP is a scripting language which can manipulate information held in a database and generate web pages afresh each time an element of content is requested from a browser. Other programs may also be included in a package, such as phpMyAdmin which provides a graphical interface for the MySQL database manager, or the alternative scripting languages Python or Perl [9]. Whatever the software model, when they are implemented, there are set of activities to be carried out from the beginning to the end of the development process.

- **Domain Analysis**

The first step in attempting design a new software piece is to investigate the domain that the problem belongs to. Assuming that the developers (including the analysts) are not sufficiently knowledgeable in the subject area of the new software, the first task is to investigate the so-called "domain" of the software.

- **Software Elements Analysis**

The most important task in creating a software product is extracting the requirements. Customers typically have an abstract idea of what they want as an end result, but not

what software should do. Incomplete, ambiguous, or even contradictory requirements are recognized by skilled and experienced software engineers at this point.

- **Requirements Analysis**

Once the general requirements are gathered from the client, an analysis of the scope of the development should be determined and clearly stated. This is often called a scope document. This document can be considered a legal document so that if there are ever disputes, any ambiguity of what was promised to the client can be clarified.

- **Specification**

Specification is the task of precisely describing the software to be written. In practice, most successful specifications are written to understand. A good way to determine whether the specifications are sufficiently precise is to have a third party review the documents making sure that the requirements are logically sound.

- **Software architecture**

The architecture of a software system is used to have an abstract representation of that system. Architecture is designed making sure the software system will meet the requirements of the product, as well as ensuring that future requirements can be addressed.

- **Implementation**

This is the part of the process where software engineers actually program the code for the project.

- **Testing**

Testing software is a very important and crucial part of the software development process. This part of the process ensures that bugs are recognized as early as possible.

- **Deployment**

After the code is appropriately tested, it is approved for release and sold.

- **Documentation**

The documenting is needed so that the internal design of software for the purpose of future maintenance and enhancement. This is required throughout development.

- **Software Training**

This is a very important phase of software development to have training sessions for those uses the system functionalities.

- **Maintenance**

Proper maintenance and enhancement is needed in software development projects as the customers requirements are changing. It is during this phase that client calls come in and you see whether your testing was extensive enough to uncover the problems before customers do.

Though there are many software processes, fundamental activities are common to all software process. Given below are the fundamental activities.

1. Software specification.
2. Software design & implementation.
3. Software validation.
4. Software evolution.

3.6 Summery

According to the applicable prevailing technology, possible software process models, analysis methodologies, design methodologies and development environment have been given in this chapter. Commonly practiced technology has been detailed comprehensively within this like waterfall model, object oriented analysis and design methodology. Comparisons of each software process models, analysis and design methodologies have been given in table form in this chapter for easy reference for the next chapter.