Chapter 3

# Technology adapted

## 3.1 Introduction

Previous chapter describes the problems and weakness of existing system, advantages of new proposed system, given comparison with proposed system and other systems available in the market. This chapter identifies technology adapting for the solution, advantages and disadvantages of each technologies

## 3.2 Software process models

To develop software of lasting quality, it is important to design a solid architectural foundation that's flexible to change. To develop software rapidly, efficiently and effectively with a minimum of software scrap and rework, we need to have the right people, the right tool and the right focus. To do all this consistently and predictably, with appreciation for the lifetime costs of the system, we must have a sound development process that can adapt to the changing needs of business technology.

There are 3 main Software process Models waterfall model, Evolutionary development model and the Component-based model.

1. The waterfall model - Separate and distinct phases of specification and development.

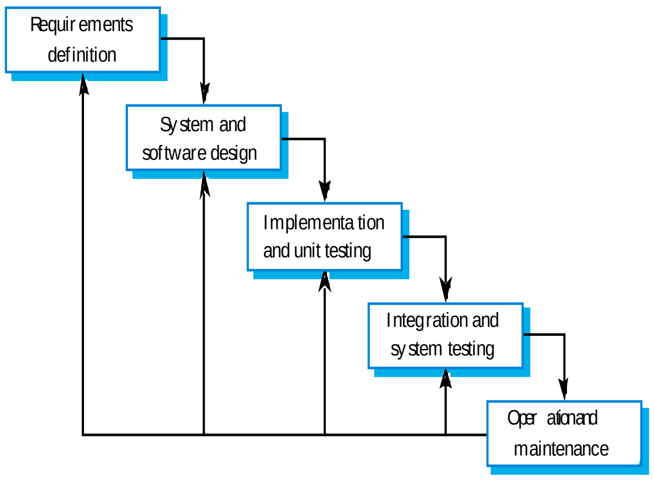

Requir ements definition
System and software design
Implementa tion and unit testing
Integration and system testing
Oper ation and maintenance

Figure 3.1 – The waterfall model

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

Waterfall model is good for large scale systems. If domain is well understood, waterfall model is used.

2. Evolutionary development

   Specification, development and validation are interleaved.

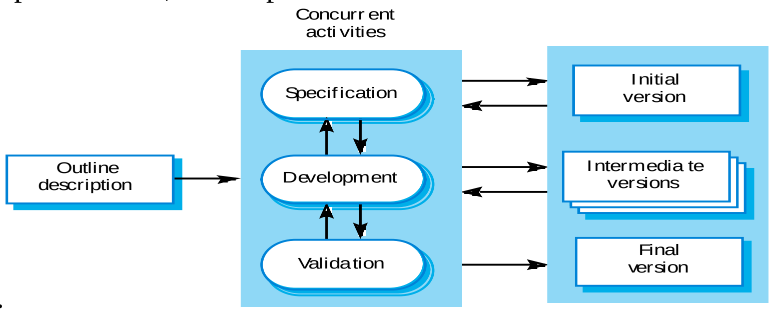


Figure 3.2 – Evolutionary development

3. Component-based software engineering

   The system is assembled from existing components

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective. [4]

### 3.2.1 Advantages and disadvantages of Software Development Methodologies

| **S/W development methodology** | **Advantages** | **Disadvantages** |
|---|---|---|
| Waterfall | • Involves a sequence of iterations of development activities.<br>• Documentation produce at each phase and that it fits with other engineering process models. Unlikely to change radically during system development<br>• Early functionality.<br>• Does not require a complete set of requirements at the onset.. | • The partitioning of the project into distinct stages is inflexible.<br>• Difficult to change customer requirements because of commitments must be made at early stages in the process<br>• Beginning with less defined general objectives may be uncomfortable for management |

Table 3.1 – waterfall model

## 3.3 Analysis and Design methodologies

Following are the Analysis and Design methodologies

1. Object-Oriented analysis and design (OOAD)
2. Structured System Analysis and Design (SSAD)
3. Agile

Object-Oriented analysis and design (OOAD) is often a part of development of large scale systems and programs often using the Unified Modeling Language (UML). OOAD applies object-modeling techniques to analyze the requirements for a context – for example, a system, a set of system modules, an organization, or a business unit- and to design a solution. Most modern object-oriented analysis and design methodologies are use case driven across requirements, design, implementation, testing and deployment.

**3.3.1 OOA:** Object-Oriented Analysis aims to model the problem domain, the problem we want to solve by developing an object-oriented system. The source of the analysis is a written requirement statements, or written use cases, UML diagrams can be used to illustrate the statements.[11] An analysis model will not take into account implementation constraints, such as concurrency, distribution, persistence or inheritance nor how the system will build. The model of a system can be divided into multiple domains each of which are separately analyzed, and represent separate business, technological or conceptual areas of interest.

The result of Object-Oriented Analysis is a description of what is to be built, using concepts and relationships between concepts, often expressed as a conceptual model.[3] Any other documentation that is needed to describe what is to be built is also included in the result of the analysis. That can include a detailed user interface mock-up document. The implementation constraints are decided during the Object-Oriented design (OOD) process.

**3.3.2 OOD:** Object-Oriented Design is an activity where the designers are looking for logical solutions to solve a problem, using objects, Object-Oriented Design takes the conceptual model that is the result of Object-Oriented Analysis, and adds implementation constraints imposed by the environment, the programming language and the chosen tools, as well as architectural assumptions chosen as basis of design. The concepts in the conceptual model are mapped to concrete classes, to abstract interfaces and their implementations for stable concepts can be made available as reusable services. Concept identifies as unstable in object orient analysis will form basis for policy classes that make decisions.[3] The result of the Object-Oriented Design is a detailed description of how the system can be built, using objects.

Traditional business applications are characterized by some or all of the following requirements.

- Large volumes of application data that is typically stored in a relational database.
- Large amounts of user interface code to give a diverse set of user access to services and data in a variety of ways.
- Large teams of analysts, programmers and testers have to be effectively managed.

- The longer elapsed time, higher efforts and cost involved in the project require the execution of the project with schedule and cost controlled on an ongoing basis.

Above problems can be solved by the usage of object orientation in the following way:

- Simplification of the real world & business entities.
- Easy management of information & high CASE tool support.
- Object & operations are easy to discuss with the user which leads to enhanced and precise requirements capture.
- Rapid application development is more easily achieved and there is a flawless changeover from one phase to the other among the development phase such as analysis, design, implementation, testing & maintenance.
- Due to this seamless transition the management of the software becomes easier than ever.
- With Object Orientation a library of objects can be developed which facilitates reusability, and conserves existing and consistent comprehension of the problem & business domain.

## 3.4 Software Modeling using Unified Modeling Language

The Unified Modeling Language (UML) has now become a standard language for writing software blueprints. UML is a language for visualizing, specifying, constructing & documenting the artifacts of a software intensive development project. The UML is appropriate for modeling systems ranging from enterprise information systems to distributed web-based applications and even to develop embedded & mission critical applications. UML is a very expressive language in which addressed all the viewpoints needed for development & deployment of such systems. [11]

The UML is process independent, although optimally it should be used in process that is Use-Case driven, architecture centric, iterative and incremental such as the Unified Software Development Process (USDP) to retrieve the maximum benefit out of it.

UML is a language for **Visualizing** –UML is a graphical language where a graphical notation is used to express the ideas rather using in a textual notation for modeling system. This facilitates ease of communication between the project stake holders & the development team. This is achieved by a collection of easy to understand symbols attached with well defined semantics. [2]

The UML address the specification of all the important phases & their decisions such as requirements specification, analysis, designing, implementation & deployment.
UML has become not only **visual modeling language** but the artifacts developed in UML can be directly connected with popular programming languages such as Java, C# and even the most popular VB.net .Not only the programming language even the UML diagrams are totally convertible into a Relational or Object Relational database Schemas.

UML is a language for **documenting** –In a software system the documentation plays a major role when it comes to communication, measurements & controlling. UML addresses all these aspects by developing various artifacts relevant for each development phase in addition to the raw executable program coding. These artifacts include requirements, architecture, design, source code, project plans, test evidence, prototypes & release. All of the above artifacts are very much important for project progress communication, measuring & estimations, controlling & monitoring activities. [2]
Following are the models used in UML to describe the various aspects of the system.

1. Use-Case Diagrams
2. Activity Diagrams
3. Sequence Diagrams
4. Class Diagrams
5. Collaboration Diagrams
6. State chart Diagrams
7. Component Diagrams
8. Package Diagrams
9. Deployment Diagrams

### 3.4.1 Use-Case Diagram

Use-case diagram (Figure 3.3) includes asset of use cases where each use case is a description of the functionality of the user's perspective. Use-case diagrams are used to show the functionality that the system will provide and to show which users will communicate with the system in some ways to use that functionality.

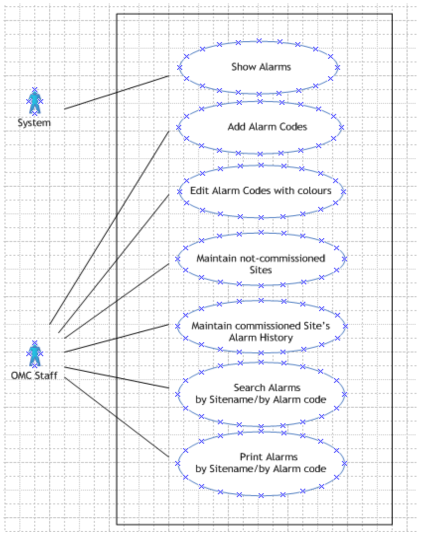
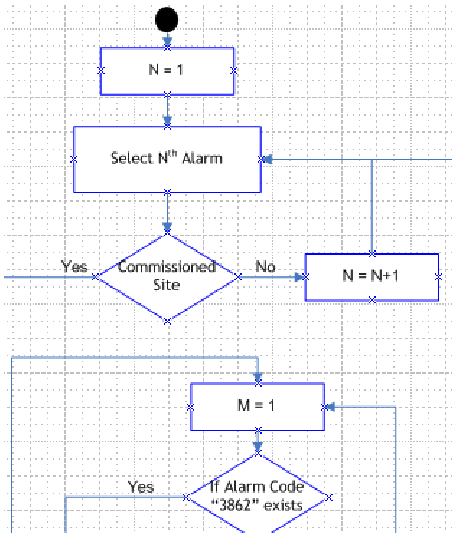

Show Alarms
Add Alarm Codes
Edit Alarm Codes with colours
Maintain not-commissioned Sites
Maintain commissioned Site's Alarm History
Search Alarms by Sitename/by Alarm code
Print Alarms by Sitename/by Alarm code
System
OMC Staff


Figure 3.3-Use-Case Diagram

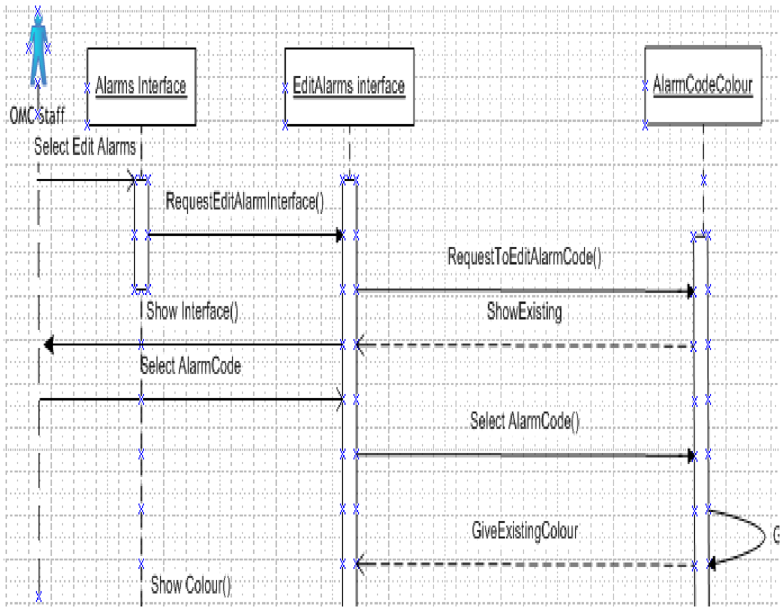
Figure 3.4- Activity Diagram

### 3.4.2 Activity Diagram

An Activity diagram (Figure 3.4) represents the business and operational step-by-step workflows of components in a system. Activity diagram shows the overall control flow.

### 3.4.3 Sequence Diagram

Sequence diagram (Figure 3.5) shows an interaction between objects arranged in a time sequence. Sequence diagrams can be drawn at different levels of details and to meet different purposes at several stages in the development life cycle. The commonest application of a Sequence diagram is to represent the detailed object interaction that occurs for one use or for one operation.



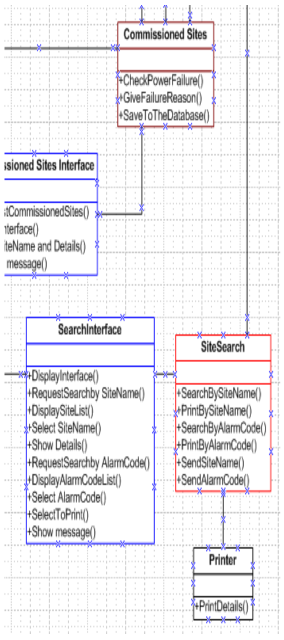


Figure 3.5- Sequence Diagram

Figure 3.6- Class Diagram

### 3.4.4 Class Diagram

The Class diagram (Figure 3.6) is a type of static structure diagram that describes the structure of a system by showing the system classes their attributes, methods and the relationships between the classes, Attributes identifies the characteristics of a class while methods identify the behavior.

Relationships are the logical links between classes and can be in different flavors.

They are Aggregation, Composition, Link, Association and Generalization.

## 3.5 Database System Architecture

### 3.5.1 ER Diagrams and Relational Tables

Entity Relationship (ER) Diagrams shows the logical database structure of the proposed future system. It also describes the entities in the database and how they relate to each other. ER diagrams are easy to understand and it is a graphical way to express the logical database structure.

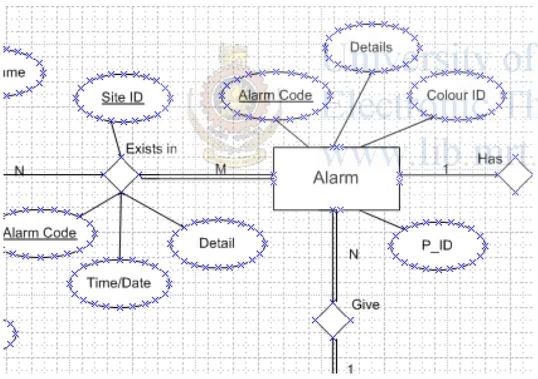


Figure 3.7- ER Diagrams

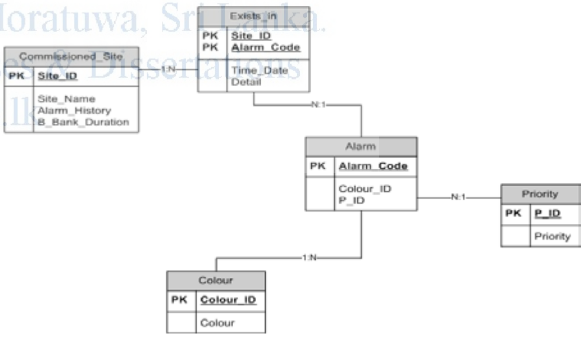


Figure 3.8- Relational Tables

## 3.6 Summary

This chapter discussed with software process models like waterfall, Evolutionary and component based. Then Analysis and design methodologies like OOAD, SSAD and Agile. After that Software Modeling Techniques using UML diagrams with aspects of UML modeling technique. Next chapter is about to selected approach, software language and scope of the project that I wish to complete.