**2. Software Development Life Cycle (SDLC)**

**Software life Cycle**: is a description of the events that occur between the birth and death of a software project inclusively.

**2.1 Stages of the Software Development Life Cycle**

The software Development Life Cycle is a series that have to be gone through to construct a system. These are:

**2.1.1 Software Requirement Analysis**

In this stage, the development team visits the customer and studies their system. They investigate the need for possible software automation in the given system. By the end of this stage, the team produces a document that holds the different specific recommendations for the candidate system. It also includes the personal assignments, cost project schedule and target dates.

**2.1.2 System Analysis and Design**

In this stage, the software development process, the software’s overall structure and its nuances are defined. A software development model is created. A analysis and Design are very crucial in the whole development cycle. Any fault in the design phase could be very expensive to solve in the later stages of the software development.

**2.1.3 Implementation (Code Generation)**

The design must be translated into machine- readable form. The code generation step performs this task. Programming tools like compiler, interpreter and debugger are used to generate the code. Difficult high level programming language like c, c++ and java are used for coding.

**2.1.4 Testing**

Once the code is generated, the software program testing begins. Different testing methodologies are available to unravel the bugs that were committed during the pervious phase.

**2.1.5 Deployment**

The software is now finished and gave to customer to operate it, one the user or may be more will execute the software in order to get his or their requirements, if any bug has been found in it, the customer must return it to development team to do maintenance on it.

**2.1.6 Maintenance**

Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system.

**2.2 Software Life Cycle Models**

A software lifecycle model is a series of steps through which the product progresses. These include requirements phase, specification phase, design phase, implementation phase, integration phase, maintenance phase, and retirement. Software Development Lifecycle Models illustrate the way you organize your activities. There are a number of Software Development Lifecycle Models, each having its strengths and weaknesses and suitable in different situations and project types.

The list of models includes the following:

**2.2.1 Build and Fix Model**

It is unfortunate that many products are developed using what is known as the build-and fix model. In this model the product is constructed without specification or any attempt at design. The developers simply build a product that is reworked as many times as necessary to satisfy the client. This model may work for small projects but is totally unsatisfactory for products of any reasonable size. The cost of build-and fix is actually far greater than the cost of properly specified and carefully designed product. Maintenance of the product can be extremely in the absence of any documentation. This model is illustrated in figure 2.1

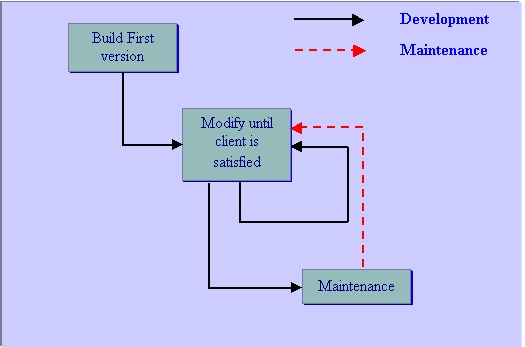


Figure 2.1: Build and fix model

**2.2.2 Waterfall Model**

The first published model of the software development process was derived from other engineering processes. Because of the cascade from one phase to another, this model is known as the waterfall model. It is also known as linear sequential model. This model is illustrated in figure 2.2.

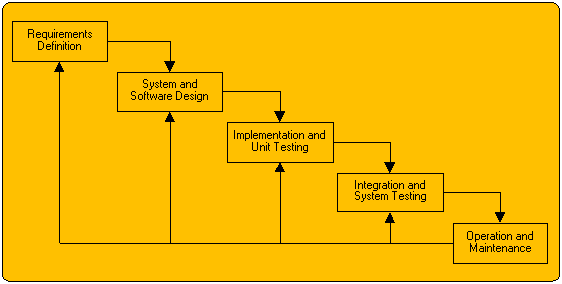


Figure 2.2 Waterfall model

It suggests a systematic, sequential approach to software development that begins at the system level and progresses through the analysis, design, coding, testing, and maintenance. The drawback of the waterfall model is the difficulty of accommodating change after the process is underway. The Waterfall model problems are:

* Inflexible partitioning of the project into distinct stages.
* This makes it difficult to respond to changing customer requirements.
* Therefore, this model is only appropriate when the requirements are well-understood.

**2.2.3 The Prototyping Model**

The Prototyping Model begins with requirements gathering. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A "quick design" then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats). The quick design leads to the construction of a prototype. This process is shown in figure 2.3. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

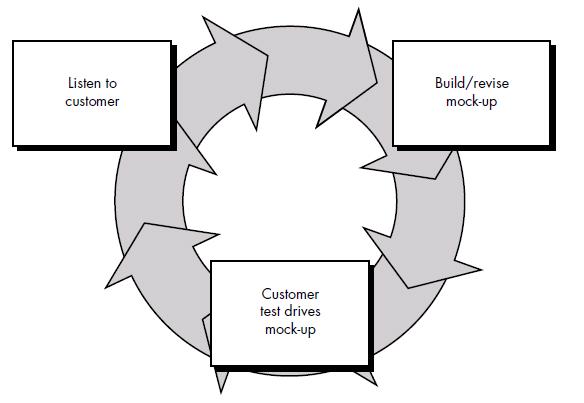


Figure 2.3 Prototyping Model

The Prototyping problems are:

1. Users treat the prototype as the solution.
2. The developer often makes implementation compromises in order to get a prototype working quickly.

**2.2.4 The Incremental Model**

The incremental model combines elements of the linear sequential model with the iterative philosophy of prototyping. Referring to Figure 2.4, the incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces a deliverable "increment" of the software. For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment can incorporate the prototyping paradigm. When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer. As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

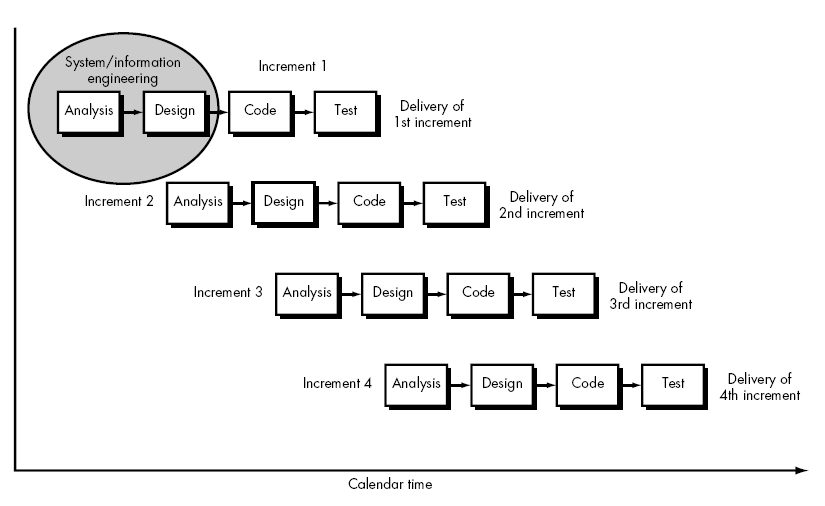


Figure 2.4 the Incremental Model