**2. PLANNING**

**2.1 Model Description**

Feasibility study

Requirement analysis and specificationspssssssspespecification

Design

Maintenance

Integration and system testing

Coding and unit testing

**Fig. Iterative Waterfall Model**

* In our project we are using iterative waterfall model.
* It is not possible to strictly follow the classical waterfall model.
* Making necessary changes to the classical waterfall model so that it becomes applicable to practical software development projects.
* The main change to the classical waterfall model is in the form of providing feedback paths from every phase to its preceding phases as shown in figure.
* The feedback paths allow for correction of the errors committed during a phase as and when these are detected in a later phases.
* For example if during testing a design error is identified then the feedback path allows the design to be reworked and the changes to be reflected in the design document.
* There is no feedback path to the feasibility stage. This means that the feasibility study errors cannot be corrected.

**Requirements analysis and specification**

* The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely
* Requirements gathering and analysis, and
* Requirements specification
* The goal of the requirements gathering activity is to collect all relevant information from the customer regarding the product to be developed. This is done to clearly understand the customer requirements so that incompleteness and inconsistencies are removed.
* The requirements analysis activity is begun by collecting all relevant data regarding the product to be developed from the users of the product and from the customer through interviews and discussions.
* During SRS activity, the user requirements are systematically organized into a Software Requirements Specification (SRS) document.

**Design**

* During the design phase the software architecture is derived from the SRS document. Two distinctly different approaches are available.
* Traditional design consists of two different activities; first a structured analysis of the requirements specification is carried out where the detailed structure of the problem is examined. During structured design, the results of structured analysis are transformed into the software design.

**Coding and unit testing (Implementation)**

* The purpose of the coding and unit testing phase of software development is to translate the software design into source code. Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested.
* Each module is unit tested for determine the correct working of all the individual modules.

**Integration and system testing**

* Integration of different modules is done once they have been coded and unit tested. During the integration and system testing phase, the modules are integrated in a planned manner.
* Finally, when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document. System testing usually consists of three different kinds of testing activities.
* α – testing: It is the system testing performed by the development team.
* β – Testing: It is the system testing performed by a friendly set of customers.
* Acceptance testing: It is the system testing performed by the customer himself after the product delivery to determine whether to accept or reject the delivered product.

**Maintenance**

* Maintenance involves performing any one or more of the following three kinds of activities:
* Correcting errors that were not discovered during the product development phase. This is called corrective maintenance.
* Improving the implementation of the system, and enhancing the functionalities of the system according to the customer’s requirements. This is called perfective maintenance.
* Porting the software to work in a new environment. For example, porting may be required to get the software to work on a new computer platform or with a new operating system. This is called adaptive maintenance.

**2.2 RISK MANAGEMENT**

**Risk Management**

* The aim of risk management is to reducing the impact of all kind of risks that might affect a project. Risk management consists of three essential activities: risk identification, risk assessment, and risk containment.

**Risk Identification**

* A software project can be affected by a large variety of risks. In order to be able to systematically identify the important risks which might affect a software project, it is necessary to categorize risks into different classes.
* The project manager can then examine which risks from each class are relevant to the project. There are three main categories of risks which can affect a software project:

**Project Risks**

* Project risks concern varies forms of budgetary, schedule, personnel, resource, and customer-related problems. An important project risk is schedule. It is very difficult to monitor and control a software project.
* It is very difficult to control something which cannot be seen.
* The invisibility of the product being developed is an important reason for many software projects failure.
* So in our project we are trying to resolve this kind of project risk which is also known as schedule risk.

**Technical Risks**

* Technical risks concern design, implementation, interfacing, testing, and maintenance problems.
* Technical risks also include ambiguous specification, incomplete specification, changing specification, technical uncertainty. Most technical risks occur due to the team member’s insufficient knowledge about the project.
* So in order to prevent this risk, we have done appropriate project analysis before starting our project.