CHAPTER – 3

SYSTEM DEVELOPMENT AND MODELING OF FRAMEWORK

For the research work to take shape, it is mandatory that the System Requirements Evaluation is to get a comprehensive and comprehensive knowledge of the system needs as described in job Source and taken within the System Situation, and to interrupt it into distinct demands, that are then obviously defined, examined and agreed upon with the Client Decision Makers. Throughout System Needs Analysis, the construction for the application is created, providing the basis for all potential design and improvement efforts. Program Requirements Evaluation could be a period, because all the major Customers and their pursuits are delivered to the procedure for identifying conditions. Because the essential kind the foundation for many potential work on-the-job, from design and development to testing and records, it is of the most significance the Project Team produces a complete and precise portrayal of all requirements the method must match. Accurately identified demands result from cooperation and successful conversation among all people of the Project Team, and supply the greatest opportunity of developing something that completely fulfills the requirements of the Clients.

The principal goal of this phase would be to generate a detailed Functional Specification defining process models and accompanying data illustrating the information to be managed, together with also defining the complete set of system capabilities to be implemented and the processes to be supported by the new system. Requirement can be categorized as Architectural, Behavioral, Functional, Non-Functional, hardware and software.
3.1 System Analysis

System analysis is a process for recommending achievable recommendations for enhancing the device functionality, identifying problems and collecting informative data, comprehend the procedures included. This requires obtaining out bottlenecks, accumulating functional data, comprehend the information flow, examining the company processes and developing options for beating the flaws of the application to be able to reach the company goals. Program Analysis additionally comprises subdividing of complex process including the recognition of manual methods; whole program and details store. To implement the system shown in Figure 3.1 this research work carried out following two modules.

2. Primary Project Management Tasks.

3.1.1 Pre –Requisite Project Management Task

These task are require before the starting the project under consideration. These tasks are important because outcome of this these tasks results in the decision of whether or not the project should be taken for development. Following pre-requisite tasks are carried out in the research work.

I. Information Gathering

Information required for the system can be collected from different research papers, websites and books. Once all require documents regarding information are collected then review of the literature can be done by extensive study.

II. Preliminary System Study

Preliminary system study is the primary stage of system development life cycle. This provides a clear image about what actually the bodily system is? It is a simple research of the machine in thought. In exercise, the first method study comprises the preparation of a 'System suggestion' which particulars the Issue Definition, Goals of the Study, Problems of research for Study, Restrictions, and
Expected advantages of the fresh method, etc. in the light of the individual needs. The original suggestion is aimed and prepared by the signing specialist.

III. Detailed System Study

The in depth investigation of the system is performed in compliance with the purpose of the system to be made. This requires depth study of different work performed and relationships within and away from system. In this working, data are gathered from the available documents, decision points and transactions used in the system. Place of observation, interviews and survey are methods used for comprehensive system study. Using these measures it becomes simple to attract the precise wall of the proposed system.

3.1.2 Primary Project Management Tasks

These tasks are the actual tasks of the project development life cycle. Primary project management tasks are listed below.

1) Concept development and preparing actual proposal
2) Analysis and Design
3) Implementation
4) Testing

I. Concept Development

The Concept Development Period might start after the conclusion of the Initiation job status review and the clearing of Concept Proposal and Project requirement along with the approval to continue the said Phase. The emphasis of the phase is: 1) Calculate feasibility of choices and 2) Specifically define and clear project scope, which includes the system, the deliverables, and all the prerequisite activities. The activities act as inputs for ITPR, which is an important output of this activity.

The success of this phase should contain:

- Definition of job scope
- Assessment of technical choices
- Formation of the job acquisition strategy
- Baseline evaluation of hazards
- Approval of job expenses and obtaining of present financial year capital
- Definition of organizing functions and responsibilities
- Development of the Work Break-down Structure.
- Development of the ITPR.
- Study and Detailed evaluation of project details to help establish the task range and Acceptance to advance towards the Planning Period

II. Analysis and Design

The analysis period identifies the requirements of the equipment, independent of how these needs will certainly be attained. This stage identifies the predicament the person is trying to solve. The result at the conclusion of the interval is a demand. The necessity file tries to recapture the demands from your client's outlook by determining aims and relationships. The evaluation team develops the demand record, which provides detail measures and concerning the issues on issues. This file must have common situations of use, states, events, and atypical scenarios of use.

In the Style Stage, the program is essentially made to fulfill the demands recognized in the prior periods. The entire requirements are recognized in the Needs Analysis Stage are changed into a Program Design Record that correctly explains the composition of the device and that may be utilized as an enter to system development next phase. The completed Design Phase should have:

- Transformation of all requirements into detailed specifications covering all aspects of the system
- Assessment and planning for security risks
- Permission to go to the Development Phase

The major purpose of the Design Phase is to convert the requirements into complete and detailed system design specifications. Once the design is approved, the Development Team begins the Development Phase.
III. Implementation

In computer science, an execution is a decision of the specification or formula for a software, applications component, or added computer system through development and installation. Several implementations may exist for a requirements or standard. For instance, web browsers contain implementations of Worldwide Web Consortium recommended specifications, and computer software development resources contain implementations of programming languages. A crucial distinction between system implementation and all other phase of the life-cycle is that all task activities around the stage have been done in safe, shielded, and safe environments, where project issues that appear to have minimum effect on day to day business procedures. However, that is not the situation, after the system goes live. It's through the careful planning, performance, and direction of system Implementation, the actions the Project Team may minimize the chance of these events, and decide proper backup plans in case of a problem.

IV. Testing

Computer software testing is an investigation performed to supply stakeholders with guidance regarding the level of the merchandise or service under assessment. Software testing too can provide an object, impartial view of the program to allow the business to identify and comprehend the risks of software delivery. The methods comprise a procedure of running a program or software with the aim of discovering mistakes or bugs within the applications.

Computer software testing is a procedure for validating and confirming about a program/application/product that:

- It fulfills the conditions as mentioned in its style and development
- It functions in an anticipated fashion
- It could be implemented with the related features.
- It fulfills the requirements of user/stakeholders
Software screening, based on the testing procedure applied, can be applied at any moment in the creation process. Nevertheless, the majority of the evaluation attempt usually happens if the code process is finished or after the conditions are described. Even though in the Agile techniques the majority of the evaluation work is, conversely, ongoing. Therefore, the strategy of the evaluation is regulated by the program improvement methodology used. The test effort will be focused by different software development models at different points in the development process. Before it reaches a proper group of evaluators, newer development versions, like Nimble, commonly apply test driven development and put a heightened part of the testing in the control of the programmer. If the programming procedure has been finished with a more conventional design, the majority of the test performance happens after the conditions are described. In several software engineering methods, the testing phase is followed after the execution another phase is finished by another group. There is value in this strategy; it's difficult to see one's own blunders, and a brand new eye may detect apparent blunders much quicker than the individual that has read and reread the content often. Unfortunately, delegating the testing activity to some other group results in a slack approach regarding quality.

3.2.1 System Requirements

- **Software Requirements**

  To develop proposed system following software tools are used:

  1. **Technology Used**: J2SE (JDK 1.5 and above)
  2. **Front End**: Java Swing Package
  3. **Development IDE**: NetBeans IDE
  4. **Analysis & Designing Tool**: Rational Rose 98
  5. **Standford Parser /Tagger**: to extract software artifacts and generation of parse tree

- **Hardware Requirements**

  - **Processor**: Intel Pentium Processor
• **Primary Memory (RAM):**
  
  It requires at least 512 MB primary memory to run system effectively. It should be good if more than 512 MB primary memory is available for system such as 1 or 2 GB RAM.
  
  • **Secondary Memory (HDD):** It requires at least 1GB secondary memory as Hard disk.

❖ **Operating System Requirements**
  
  • Windows XP/ Windows Vista/ Windows 7 Home or business Edition

### 3.3 Architectural Requirement

- The System components should work with in accordance with each other.
- All Components should be placed in one architectural environment.
- All the components should be interlinked according to their association.

### 3.4 Behavioral Requirement

- All the components should coordinate their work properly.
- Modules of the system should invoke appropriate functions
- User should be provided with easy Interface.
- No complexity for the handling system should be provided to user.

### 3.5 Functional Requirements

In software engineering, a functional demand identifies a function of the software system or its part. A function is called a group of the behaviour, inputs, and results. Practical requirements specify special effects of the method, as described in requirements engineering.

Following are the functional requirements of the developed framework.

1. Generation of SBVR format text to normalize the Natural language Requirement
2. Generation of Use Case Model
3. Generation Complete Class Model
4. Generation of OCL
5. Providing XMI Support
6. Identify the candidate classes, candidate methods, candidate attributes
7. Extract a Core Prototype Model
8. Develop Complete-Meta-Model (Complete class diagram) from source code
9. Use of UMLsec annotations
10. Extraction of relationships and secure attributes from generated models
12. Metrics Analysis.

3.6 Non Functional Requirement

Non-functional necessity is really a necessity that identifies standards that may be utilized to judge the procedure of the system, rather than specific behaviours. This must be compared with practical demands that explain special behaviour or functions. The strategy for applying functional needs is comprehensive in the machine layout. The strategy for implementing nonfunctional needs is comprehensive in the device structure. Nonfunctional requirements are commonly called qualities of the method. The non-practical requirements for the research work are mentioned below.

1. **Usability:**
   
   Proposed system would be very user friendly. The user does not have to worry about the internal details of the proposed techniques. The user directly use the functions provided by it on various problem statement from different domains. The system should be easy to follow.

2. **Reliability:**
   
   The system should be very reliable and produce very good output. It should generate output with high precision and recall values than all existing tools and techniques.
3. **Availability:**

   The system should be available 100% to the user and it should be operational and used round the clock throughout the year.

4. **Cost:**

   The system should be developed with minimum cost i.e. it requires less money to develop. To fulfil this requirement system should be developed with freely available tools and techniques.

5. **Performance:**

   Performance of proposed approach should be good due to use of the intermediate SBVR format. Such SBVR format will nullify the inaccuracy introduced due to ambiguous and inconsistent nature of natural languages.

3.7 **Feasibility Study**

   It is really a check of system suggestion based on its execution, effect to the business; create what user wants and an efficient use of assets. It's essentially the evaluation of the practicability of a proposition of the job. This is actually the most significant feature in system analysis and design. An initial study is performed before the actual function of a job begins to assess the chance of the projects achievement. Investigation is completed to decide whether to proceed with a project that's under consideration, centered on such variables as the competitors, the market, accessible technology, work force, and money. By examining this, direction may take decision about the system. This really may be the cost benefit evaluation. The measures that are completed for feasibility research are the following.

   - Analyze the planned undertaking and create a created explanation.
   - Define and record potential kinds of techniques.
   - Develop declaration of the kinds of program
   - Analyze the prices of comparable systems.
   - Produce a rough approximation of the device size, prices and schedules.
• Define the advantage of the system.
• Include both quantitative and qualitative steps.
• Produce an estimation of another life cycle period.
• Present feasibility report to customer for discussion.

The aim of feasibility research is not just to resolve the issue but to get a feeling of its extent. The three concerns required in feasibility evaluation are described here.

a) Economic Feasibility

Economic evaluation may be probably the most often used approach for assessing the efficacy of the prospect system. It's a process, which establishes the conserving and advantages that are required from something and compare them against the prices. When the advantages outweigh the expenses, then the machine must be created and applied. This factor is related to the cost of the system. Various costs are taken into consideration while evaluating the effectiveness of the system.

b) Technical Feasibility

This feasibility involves financial consideration to install the required hardware and software. It means deciding whether the proposed system is technically feasible or not. At present the system is entirely manual. Necessary software and hardware is to be installed. The proposed system is developed with java2 platform, hence the project is technically feasible.

c) Operational Feasibility

It considers the acceptability of the system. It checks whether the system will be put to proper use if developed and implemented. It also checks whether the client are able to handle the system, whether the proposed system will cause any trouble to client. The interface is present to its users and the users interact with the application through windowing system. The system developed is requiring no extra technical knowledge for the end user. Hence the system is operationally feasible.
3.8 Risk Management

Actually “Risks are future undecided events with a probability of occurrence and a potential for loss”.

Risk management is the overall package of recognition, evaluation and prioritization of risks. In proposed system risks can be occurred from hesitation in financial markets, project failures (at any phase in design, development, implementation, testing and deployment), legal liabilities, credit risk etc. Therefore risk identification, assessment and prioritization of proposed system is necessary to reduce such risk for better software system. Usually the schemes to handle risk contain moving the risk to another party, averting the risk, decreasing the damaging effect or chance of the risk, or even taking some or all the possible or real outcomes of a special risk. Clear evaluation of software may help effective preparation and projects of function.

The term risk is defined as the potential future harm that may arise due to some present actions. Risk management in software engineering is related to the various future harms that could be possible on the software due to some minor or non-noticeable mistakes in software development project or process. Risk management is the most important issue involved in the software project development. The risks for research work are categorized in the categories given next.

3.8.1 Risk Identification

Risks are documented, secret and handled before actual deployment of program. Risks can be identified and classified for proposed system.

Schedule Risk

Proposed software schedule fails if project tasks and schedule release risks are unattended. They are largely change on applications market and eventually on project and might cause project failure. Schedules fail due to following reasons:
Wrong Time Estimation

If assets and resources like staff, systems and skills of persons are not monitored correctly, then it will lead to Failure to determine complicated functionalities and time will be needed to build up these benefits which will result in sudden job scope expansions.

Operational Risks

In this the loss occurs due to not right process deployment and unperforming system or outside events risks. This type of risk directly affected on software and subsequently economy of software and it causes for project failure. Therefore such risk must eliminate. The main development for this is due to failure to solve major conflicts, not clear the assignment, scarce resources, no training, no material planning, no interaction in team. Therefore such risk can be eliminated by make proper plan for priority order for operation, responsibilities of it, sufficient amount of resources, training as per requirement and proper communication.

Technical Risks

This risk directs failure of functionality and performance of the proposed software project.

Causes of specialized risks could be constant shifting demands, aged technologies accessible or the present technology is really in first periods, merchandise is complicated to apply, challenging endeavor modules integration. Therefore such risk can be eliminated by properly understand of requirements and their changing requirement as project progress, build plan for project modules integration and regularly examine the integrated for causing new risk or not.

Analysis and Design Risks

The analysis phase risk is minimized by exhaustive study of literature available and the solution is planed by conducting meetings, discussions for every possibility of the proposed system. During the Design Phase, it is designed to satisfy the requirements identified in the previous phases. The requirements
identified in the Requirements Analysis Phase are transformed into a System Design Document that accurately describes the design of the system and that can be used as an input to system development in the next phase. The risks in the design phase are minimized by carefully and accurately designing the system.

**Implementation Risks**

The Implementation phase has one essential action: implementing the new system in its main atmosphere. Supporting actions include training end users and getting ready to change the system around to maintenance employees. The danger in implementation stage is reduced by doing risk management activities throughout the implementation stage. These routines include risk analysis, recognition, response preparation and observation and control.

**Testing Phase Risks**

The Testing Phase concentrates on an empirical analysis where the outcomes describe the essence of the system. Testing cannot support a system functions correctly under all the problems but may create that if it fails under certain problems. The earlier a flaw is discovered in the improvement process the more affordable the repair. Examining early in the system life cycle reduces hazards for example routine delays or price overruns. There is possibility of risks in testing phase if the quality assurance policies are not implemented effectively while testing the system. The risks in the testing phase are minimized by quality assurance policies effectively for every module of the system. Regression testing is also implemented to minimize the risk of any bug in the system.

**3.9 Planning and Scheduling**

The secret to a successful project is in the planning. Developing a project plan is the very first factor one must do when undertaking any sort of project. The project plan provides quality on expectations for the project and each deliverable in it. A formal, approved document is used to guide both project performance and project control. The main uses of the task plan are to document planning suppositions and decisions, ease communication among stakeholders and document approved scope, price, and schedule baseline.
Project Scheduling

It is an work that divides the calculated effort across the scheduled duration by giving due effort to main task. The various sets of task and their status is determined as below.

1. Determining the Scope of project and Problem definition

The scope of the project under consideration and detailed problem definition is defined in this section.

STATUS: Problem Statement and scope are defined and well understood.

2. Performing Analysis and Determining Software Requirements

The analysis of the project is done. The software requirements and functional requirements are also analyzed and a document is prepared. The required resources are then collected. In this stage a project plan is developed and is reviewed by the internal Project Guide.

STATUS: The analysis phase is complete.

3. Design

Design is a multi-step process, which incorporates many activities need to perform in project development. These activities includes identifying the different operations like developing the functional specifications, developing a prototype, obtaining feedback to the specifications and design of the system as well as database design, managing cluster, play media file etc.

STATUS: Design phase is completed as per the requirements.

4. Development

The development phase includes the activities like identifying the various modules and design parameters, developing the code and developing the test case.

STATUS: The Development phase is complete.

5. Testing

The testing phase includes applying of various testing strategies for testing the software. The test cases are developed to conduct testing. The code is tested for number of times.

STATUS: Testing phase is complete.
6. Prepare Report

Project report is generated after successful implementation and testing of the system and necessary changes are incorporated.

STATUS: Documentation phase is complete.

3. Deployment and Post Implementation review

This includes activities like deploying the software and creating a software maintenance structure.

STATUS: Software is ready for use.

3.10 System Modeling and Design

System modeling is a technique to express, visualize, analyze and transform the structure of the system. Here, a method may consist of software components, hardware components, or both and the connections between these components. Something model is then a skeletal model of the device. System modeling is meant to help in developing and maintaining large systems with focus on the construction phase.

System modeling can increase reliability and decrease development expense by making it easier to build systems, to re-use previous built parts within new methods, to change methods to satisfy changing requirements like platform changes and practical improvement, and to understand methods. Systems design involves identifying the structure, parts, modules, interfaces, and information for a system to meet specified requirements. It can be viewed it while the use of systems theory to item development. There's some overlap with the professions of systems analysis, systems structures and systems engineering. Systems design is consequently the procedure of developing and identifying systems to meet specified requirements of the user.

OOAD methods are getting to be the most commonly used methods for personal computers style. UML is a standard and popular modeling language in the area of object oriented software engineering.
3.11 Use Case Diagrams

The Use Case Diagrams in UML are useful for modeling the energetic aspects of the system. The Use case diagrams are used for modeling the behaviour of the system, a subsystem or a group. They present a set of use cases and actors and connections between them. The use case diagrams include use cases, actors, dependency, generalization, system boundaries, packages and association relationships. Use case Diagrams are used for modeling the demands of the system and modeling the framework.

![Use Case Diagram for Module-I](image)

**Fig. 3.1 Use Case Diagram for Module-I**
Use Case description

Use case 1: NL to SBVR Conversion

**Purpose:** Conversion of Natural Language Input to SBVR Format

**Primary Actor:** User who uses application/ Software Analyst

**Flow of Event:**
- User browse the operating system File Browser
- System take input of selected file
- Input file is given to System

**Post Condition:** Conversion Take Place

Use case 2: Tagging

**Purpose:** To tag the words with basic POS Tags

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**
- System scans converted input
- POS Tagging Take place

**Post Condition:** Words are tagged with Appropriate POS Tags

Use case 3: Lexical Analysis / Morphological Analysis

**Purpose:** To do Morphological Analysis on tagged input

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**
- System scans Tagged Input
- Removes the Suffixes Attached to Noun Phrases
- Removes the Suffixes Attached to Verb Phrases

**Post Condition:** get words with their original form

Use case 4: Parse Tree Generation

**Purpose:** To generate a parse tree for pos tagged output

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**
• Parser scans Tagged output
• Apply the rules
• Parse tree generated

**Post Condition:** Parse tree for input is generated

**Use case 5:** Generate use case model

**Purpose:** To generate a use case model from parse tree

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**

• System takes the morphologically analyzed input
• Actor and use cases are identified
• Relationships between actor and use case are established from parse tree

**Post Condition:** Use case model is generated

**Use case 6:** SBVR Extraction

**Purpose:** To perform Role labeling and Concept identification

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**

• Proper nouns are identified as Individual Concepts
• Common Nouns are identified as Noun Concepts
• All action verbs are identified as verb concepts
• Auxiliary verbs are identified as Fact types
• Possessed Nouns are identified as attributes

**Post Condition:** All SBVR Concepts are identified

**Use case 7:** Information Mapping

**Purpose:** To perform Mapping of SBVR Concepts to Software Artifacts

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**

• Noun Concepts Tends to be classes in class model
• Verb Concepts tends to be Method of classes
• Characteristics tends to be data items of classes

**Post Condition:** All possible elements for class model are mapped

**Use case 8:** Class model generation

**Purpose:** To generate a class model

**Primary Actor:** User who uses application / Software Analyst

**Flow of Event:**
- Collect the mapped information
- Accordingly generate the class model

**Post Condition:** class model is generated

**Use case 9:** XMI file generation

**Purpose:** To generate a XMI file

**Primary Actor:** User who uses application / Software Analyst and UML tool

**Flow of Event:**
- Collect information for conversion
- Convert the Use case information in to XMI Form
- Convert the Class Diagram information in to XMI Form

**Post Condition:** XMI File with model information is generated

**Use case 10:** Visualize the models

**Purpose:** To visualize the models

**Primary Actor:** User who uses application / Software Analyst and UML tool

**Flow of Event:**
- Take input as XMI File
- Submit Use case model XMI file to UML Tool
- Submit Class model XMI file to UML Tool

**Post Condition:** Both the models are visualized in the UML Tool Editor
Fig. 3.2 Use Case Diagram for Module -II
Fig 3.3 Use case Diagram for Module -III
3.12 Activity Diagrams

Activity is a parameterized behavior represented as coordinated flow of actions. The stream of execution is modeled as activity nodes linked by activity edges. A node could be the delivery of the subordinate behaviour, like an arithmetic calculations, a call to an operation, or manipulation of object contents. Activity nodes also include stream of control constructs, for example concurrency control, and synchronization, choice. In an object oriented design, activities are typically invoked indirectly as techniques bound to procedures that are immediately invoked. Activity features activity nodes which could be action, item or control. The proposed system is having activity diagram as shown in figure 3.4

Fig. 3.4 Activity Diagram for Extraction of Models from Natural Language Requirement
Fig. 3.5 Activity Diagram for Re-engineering & Extraction of Software Quality Metrics
Fig 3.6: Activity Diagram for Identification of validated Secure Design for Quality Assurance
3.13 Sequence Diagrams

A sequence diagram can chart a situation explained by a use situation in step by step detail to determine how things collaborate to reach your program's aims. A lifeline in a series diagram signifies an action and exhibits all its factors of conversation with extra items in events that are essential to it. Lifelines descends vertically to symbolize the passing of time and begin within the peak of the sequence strategy. Relationships between items, communications and solutions are powered as level direction arrows joining lifelines. Figure 3.7 shows the sequence diagram of the system. In this sequence diagram there are seven entities on horizontal axis objects ‘Analyst’, ‘NL Convertor’, ‘Analyzer’, ‘Extractor’, ‘Mapper’, ‘XMI Generator’, and ‘UML tool’. Dashed lines indicate that life time of the object. Rectangle on dashed line shows the object activation. Message from one object to other is shown by the dark arrow.

Sequential order interaction between objects shows into following Figure 3.8: Sequence Diagram for Re-engineering & Extraction of Software Quality Metrics
Design and Develop an Environment to Analyse Object-Oriented Software and Quality Assurance

Submit the source code represent into .class file to analyzer which parses the .class file and holds all information about the classes regarding .class source file.

1. Metrics Extractor can be applied rules for software quality metrics for assessment and extraction on analyzer.

2. All extracted software quality metrics are visualized on visualization screen.

3. Also visualizing class diagram as design view of system is extracted from class metrics and analyzer and display on screen as design view.

Fig. 3.8 Sequence Diagram for Re-engineering & Extraction of Software Quality Metrics
Fig. 3.9 Sequence Diagram for Identification of validated Secure Design for Quality Assurance
In figure 3.9 five objects are used namely application IDE, Input extractor, Genetic Optimizer, Metrics Calculator and Visualizer. In this it can be inferred that

- Application IDE provides interface to communicate with the system.
- Input extractor extracts input from user which is nothing but the class diagram. Genetic Optimizer optimizes design to produce alternate design.
- Metrics calculator applies metrics on design and code. Compare that metrics result and then generate result.
- Visualizer visualizes result in the form of graphs.