Chapter II: SOFTWARE ENGINEERING TOOLS

This chapter described the overall techniques and methods of software engineering as well as the concepts of Information technology system with respect to our research objectives. We have initially discussed in this chapter, the significant role of IT system in various domain. Studied the importance of Strategic Information System (SIS), components of IT which supports to increase performance in the organization. Here, explained the IT capabilities and their organizational impact. We have referred the basic theory of foundation of Software Engineering concepts. How it is important to analyze the basic foundations of the subject and concern with research domain. What are the hidden techniques to build the various software products? By understanding the basic concept of subject, it was helpful to clear the research objectives. Various Software Life Cycle Models (SDLC) and its significant characteristics, software crises, the remedies of software, concepts of software testing, CMM (Capability Maturity Models), etc has been studied and explained in this chapter. We have finalized various tools, techniques, and understood the theoretical conceptualization with reference to our goals. Following are various relevant significant theories which have supported to make research objectives fulfilled.

2.1 Technical System

The term Information Technology arises on large scale on the decades of 1970’s. Due to heavy burden of data available for processing, people were looking for the machine support to process the data accurately, efficiently and in less time. They found the traditional way inconvenient for the large mass of data. For fulfillment of the necessity there was an arrival of computer in each and every sector of life. Computer technology system contains, the study, design, creation, utilization, and management of computer machine based system.

People used information technological tools to manage and process the information. Atomization process use in financial sector for transaction system. This type of working methodology is used in the financial Institute since long years. For that purpose we are going to utilize software engineering model based techniques for theoretical evaluation of atomization process.
2.1.1 System operation
The computer industry continues to grow at record speeds. Computer shipments exceeded 52 billion in 1983, and the rapid growth in personal computers sales will push this number still higher each year. Well over a million people now work in the computer industry, and this does not encompass the millions who work with computers indirectly, including bank clerks who put all their transactions into computers, airlines and motel employees who work with computers to make reservations, and machines who use computer-controlled power tools. The business machine performs only a few calculations using each datum, a great volume of a data must be processed [78].

2.2 The IT capabilities and their organizational impacts are briefly illustrated hereafter [44]: Table 01: IT capabilities and their organizational impacts

<table>
<thead>
<tr>
<th>Organizational Impact/ Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Technology has various capabilities structure like Tracking, Sequential, Knowledge Management, Informational, Geographical, Transactional etc.</td>
</tr>
</tbody>
</table>

With these capabilities each are built their own organizational advantages and precious impact like the detailed tracking of task, status, inputs and outputs, different task to be processed simultaneously.

The capability of Information Technology indicates betterment of applications in the various systems. To reduce the complexity and human work process

The following system has been studied for understanding the meticulous concept for the financial organizational behavior level for facing organizational competitions and achieving the goals.

In the competitive strategy, the system like Strategic Information System (SISs) provides the precision shape of an organization. It supports to enhance productivity and performance for achieving the target of an organization

2.2.1 Strategic Systems: Role in IT

Information Technology provides several ways in strategic systems. It provides reengineering concepts, competitive intelligence, environmental changes, and technological innovations etc. [26].

The main activities of EDP (Electronic Data Processing) Management covered are [65]:

1. Planning for I.T. Function
2. Planning for I.T. Site
3. I.T. Personnel
4. Procurement of Hardware and Software
5. I.T. Operations – Procedure and controls
6. System Implementations and post implementations Review
7. Security and
8. I.T. Audit

The following important and relevant literature of the software which is one of the key elements of our research process. The following literature shows its characteristics, classes, attributes, etc.

**Significant of Software**

The nature and complexity of software has been changed significantly in the last four decades. When the computer age started, software has assumed strategic importance in information technology. Now it uses in all sectors. Especially, it provides facility to the discipline like information management and decision-making [82].

**2.3 Characteristics of Software**

It is significant to check the software characteristics that make things different. The human creation process such as construction, analysis, design, testing translated into hardware just as physical form. Software system relevant to the logical form. Developed or Engineered concept use for software product and not use manufactured concept in the classical sense

1. Software is unit based
2. As per the requirement of client.

**2.3.1 Classes of software**

- Bug free execution & reliable
- Produces correct results
- Reusable structure
- Easy to maintain
- Efficient use of computing re-sources
- Easy to understand
• User friendly

2.3.2 Attributes of Good software

Maintainability

The ease of maintenance of software is very important and critical to both the software engineer and its user. If the changes are not quickly effected, the execution of business process will be disturbed.

Dependability

Dependability is the result of a number of sub-attributes, namely: reliability through assured performance fully verified and validated processes for all functionality and features, secured and safe to work in all eventualities of hold up, breakdown and power crisis. It is dependable in all conditions and situations.

Efficiency

The software is termed good if it uses the resources at its command in the most effective manner. The resources are memory, processor and storage. The software design and architecture should be such that it offers quick response in the least processing time, using resources at optimum level.

Usability

Software becomes usable if it does not call for extra effort to be learnt. Usability increases with good documentation. User interface techniques and efficiency of user manual are the important concept in software development methodology.

Types of Software

Software as defined has several applications in many areas. As software application is possible where procedural steps are involved, and where these steps are logically linked to each other. Most software’s deal with input of multiple data types processes it by predefined rules, validate the results and produce an output for some purpose.

• System software
• Business software
• Design/Engineering/ Scientific Software
• Embedded software
• Artificial Intelligence (AI) Software
Basic objectives of software engineering are:

- **a) Consistency**: Produce a software with consistent quality
- **b) Low cost**: Produce a software with lesser cost
- **c) High Quality**: Produce a software with higher quality
- **d) Small Cycle Time**: Produce a software in a shorter time
- **e) Scalability**: Produce a software with large size and used by more number of Users [69].

Followings are the points from which software crisis are evolved

- Software products are difficult to alter
- Time and budget
- Difficult to debug and enhance
- Fail to meet user Requirements
- Larger program size
- Skill shortage
- Low productivity improvement
- Lack of adequate training software engineering

To overcome the software crisis the remedy is to spread software engineering practices among the engineers. The term Software Engineering was first introduced at the NATO conference in Germany to discuss the ‘Software Crisis”. The basic aim of software engineering is to develop quality software that is delivered on time, within budget and in consonance with the specified requirements.

People can also refer the W$^5$ HH Principle designed by Barry Boehm for betterment of system development.

### 2.4 W$^5$ HH Principle

$\Rightarrow$ **W$^5$ HH Principle**: According to Barry Boehm [BOE96] states: “you need an organizing principle that scales down to provide simple [project] plans for simple projects.” Boehm produces new approach regarding project features and relevant sources. He described new approach in WWWW- HH principle. With the support of prepared questionnaire, he describes the key project characteristics[5].
2.5 Role of Software Engineering

The term ‘Engineering’ refers to the application of scientific principles in the design, manufacture and operation of structure and machines [30]. Now a day in a single click of mouse we can execute the desired software. The working objective of the software is to be executed as per the user requirements. Software, It is sequential set of instructions to produce output after performing the operations on input. It includes a set of documents like operations and system manuals, implementation, installation manuals. Earlier, people did the computer programs by their own intuitions. By achieving the quality and maintenance of software product, it needs to consider the significance parameter like cost and time. We have to concentrate in a disciplined manner on both the quality of the product and on the process used to develop the product to achieve the objective.

A framework which includes structure, plan and controlling parameter which refers for software development methodology. With this framework we can develop information system. This type of frameworks has its own characteristics. One methodology is not suitable for different type of projects. The project should be accepted comfortable techniques which contains technical parameters and organization people. Various methods, models and tools, are incorporated in the process of basic system of development of software product. The methodology should be frequently documented in some kind of formal documentation. The framework bound to the organizations objective for development, support and cooperates the relevant methodology.

2.5.1 Software Engineering practice

We introduced a generic software process models composed of a set of activities that establish a framework for software engineering practice. In this basic system contains activities like transaction of messages, modeling, planning etc. These activities based for construction of software development process. Various activities in framework describe the proper architecture for software development discipline. The activities like planning, deployment etc. and models are included in a frame work. But how does the practice of software engineering fit in? In the sections that follow, we consider the generic concepts and principles that apply to frame work activities.
The Essence of Practice

George Polya [Pol45] outlined the essence of a problem solving, and consequently, the essence of software engineering practice:

**Understand the problem**
- For understanding the problem various parameters should be considered relevant to feature, functions behavior and data’s to obtain for exact solution of problem. It is easy to understand the problem if it is graphically represented.

**Plan the solution**
- After understanding the problem to make a plan properly to execute the solution.
- Verify the existing problem is similar to earlier and defined sub problem, find solution of sub problem. Check the design model is created or not.

**Carry out the plan**
- Check the solution as per the plan and source code as per design model. Verify the component part which should be correct in desired level.

**Examine the result**
- Confirm the procedure can be carried out for testing component part. Check the possible testing sources executed.

2.6 Foundation of Software Engineering

The most of work of human being is done by the software. The software made by the software engineers. The engineers have a wide knowledge to solve the problem efficiently and resolve the complex problem by using software methodology techniques. Software carried out most of the significant task with their appropriate goal.

The earlier software has been failed due to lack of sophistication procedure. Implementation of program was differing in the execution of application from what the customer exactly desired. After the availability of faster machine and programming languages, software engineers systematically arranged past innovative experience of writing quality of program by using engineering approach. The techniques of building software product have been developed in a systematic, cost effective and well discipline
manner. Therefore, it can be useful other than the software product development and useful for doing research in external field application [69].

2.6.1 Emergence of Software Engineering

Past accumulated experience of writing programs and innovative ideas makes techniques for Software engineering. The following are the significant points that have majorly contributed in the discipline of software engineering. These points have important role for development of software engineering discipline.

A. Early Computer Programming
B. Programming Languages
C. Design of Control based
D. Design of Structure-Oriented
E. Design of Data Flow-Oriented
F. Object Oriented Design [67].

2.6.2 Ethical issues

Software products are developed and maintained by humans. If those individuals are hard working, intelligent, sensible, up to date, and above all, ethical, then the chance are good that the way that the software products they develop and maintain will be satisfactory. Unfortunately, the converse is equally true.

Most societies for professionals have a code of ethics to which all its members must adhere. The two major societies for computer professionals, the Association for Computing Machinery (ACM) and the computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS) jointly approved the Software Engineering code of Ethic and Professional Practice as the standard for teaching and practicing software engineering [IEEE/ACM, 1999]. It is lengthy, so a short version, consisting of a preamble and eight principles, who also produced. Here is the short version:

The IEEE-CS/ACM Joint Task Force recommended Software Engineering Code of Ethics and Professional Practice (Version 5.2)

Software engineers developed the software product by using engineering techniques. It includes analysis, design, coding, implementation, testing, and maintenance. With reference to respected and beneficial profession of software product development and
welfare of the society, people working under software development process shall observe the below mentioned principles.

- Interest of public
- Client and employer
- Use highest professional standard
- In the judgment, contain independence and integrity
- Ethical approach towards management
- Maintain reputation and integrity
- Software engineers shall be supportive
- Ethical approach towards profession

2.6.3 Software Engineering Techniques

Software Engineering has emerged as a discipline like the civil, mechanical and electrical engineering, where engineering principles such as modeling, prototyping, designing, developing, testing, delivery, installation and maintenance are the broad steps woven together to achieve the final engineering objective. While these engineering disciplines are more or less a science, based on mathematics, material sciences, physics and chemistry, the same is not true for software engineering. Software engineering tends to be closer to science essentially because of its approach being scientific, systematic and the use of standards, protocols tools and techniques. Further; it encourages the principles of good and architecture. In view of this software engineering at the core level is an engineering science.

The software engineering discipline is maturing day by day. Now a day, in society increases demands of software product because of its dependence increases in all corner of life. We can estimate development time and cost with reasonable accuracy and precision. Software products are prepared very carefully by considering all relevant parameters. These software used for most of the trusted and sensitive areas and it works properly. Software engineers prepare a software product by systematic way, well disciplined manner and cost effectively. This study approaches the software engineering techniques are lucrative when researchers carried out their research work since inception to end result in any domain. It requires various types of planning so that researchers will achieve their Objective. In software Engineering Software Development Life Cycle
(SDLC) models are available and other relevant techniques use for development of software product. Researchers can refer SDLC models and other software engineering techniques which can be cooperative as role models for the research work.

2.6.4 A process framework

Apart from that a set of umbrella activities are used for the overall process of development of software product [69]. With reference to figure 4, the activity of framework related to the software engineering discipline. Similar type of task is produced for work product in development process. The individual task is performed by the action. The process of framework defined below is applicable for the software development projects.

![Software process framework](image-url)

**Figure 4: A software process framework**
**Communication** – This activity deals with overall communication, requirements gathering with customer and other stakeholders.

**Planning** – It contains work plan that should be obey. Technical parameter, Schedule, risk etc. are the parameters under planning.

**Modeling** – To produce relevant models for both customers and developers including designing, requirement with respect to development.

**Construction** – Creation of exact coding and testing for further processing steps.

**Deployment** – The product is handover to the client who has checked it thoroughly and supply the feedback depend on its product examination.

### 2.7 History of Development Models

The objective of development models to produce software product that is having suitable characteristics. It should be possible to determine in advance how much time and effort will be required to produce the final product. Software development processes observe the path of software development life cycle for software product building.

Life cycle of the software process starts from concepts exploration and ends at the retirement of the software. Software Development Life Cycle (SDLC) is nothing but a process of creating and maintaining a good software product. Various process are carried out during life cycle phase. The significant objective of SDLC is to deliver an application that fulfills the customer requirements. This can be achieved through a systematic manner, step-wise and cost effective way called the software development models. A software development process usually begins with a feasibility study. After completion of feasibility study then focus on requirements analysis and specification. Then to start the work on various phases specified in the life cycle model. It should necessary to choose the model based on the project and follow the steps of selected model. Precise communication and understanding should require during the developmental stages of software product, otherwise it may produce chaos and project failure. Every phase of life cycle model defines entry and exit criteria.

A phase is to be completed only if the respective exit criteria satisfied. Similarly, a phase can start only if the corresponding phase- entry criteria are satisfied. For example, the phase-exit criteria for the software requirement specification phase can be that the Software Requirement Specification (SRS) document has been completed.
internally reviewed and approved by the customer. Software project managers achieve the task of the project and obtain the information due to well-defined entry and exit criteria for various phases. Thus, a major advantage of adhering to a well-defined life cycle model is that it helps control and systematically organize various activities. In other words, life cycle models encourage development of software in a systematic and disciplined manner.

When a life cycle model is adhered to, the project managers can easily tell-at which stage (e.g. design, code, test, etc.) of development the project currently is. If no life cycle model is adhered to, it becomes very difficult to chart the progress of the project. [67].

In the IEEE standard Glossary of software Engineering Terminology, the software life cycle is:

“The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirement phase, design phase, implementation phase, test phase, installation and check out phase, operation and maintenance phase, and sometimes retirement phase”.

A software life cycle model is a particular abstraction that represents a software life cycle. A software life cycle model is often called a software development life cycle (SDLC).

Most business organization carry out their business through some sequence of well defined steps (called business process) similarly, manufacturing industries follow certain steps to develop their product (called manufacturing process). Since a software life cycle has somewhat similar implications with regard to software development, a software life cycle model is often referred to as software process model. The benefits of follow the path of selected model is that, to provide motivation for product development in engineering techniques.

List of various Software Life Cycle (SDLC) Models

- Waterfall
- Incremental
- Spiral
- Prototyping
• Evolutionary
• Object-oriented
• Embedded

2.7.1 Why Document a Life Cycle Model

A life cycle model forms a common understanding of the activities among the software engineers and helps to develop software in a systematic and disciplined manner.

⇒ To avoid misinterpretations
⇒ To identify inconsistencies
⇒ To avoid redundancies
⇒ To enhance the understanding of the process
⇒ To accurately define every activity

A documented life cycle model besides presenting the misinterpretations that occur when the life cycle model is not adequately documented also helps in identifying inconsistencies, redundancies and omission in the development process. Documented life cycle models are that it enhances the understanding of the process among the developers and mandates the software development organization to accurately define every activity in the life cycle. Software development model like Waterfall model follow the precious way for development of software. It includes, study for solutions, strategy, analysis and requirement specifications for the developmental software product with the help of construction and deployment. In the software engineering model, Waterfall model is the oldest model.

In summary, where the requirements are easy to establish and are stable, the development is customer specific, and changes are not foreseen in the near future, development of software through the ‘Waterfall Model’ is recommended.
2.8 Reference Parameters of Software Development Life Cycle (SDLC)

The model involves the following phases –
- System study
- Feasibility Study
- Requirement Analysis and Specifications
- Design
- Coding and Unit Testing
- Integration and System Testing
- Maintenance
System study phases passes through the following steps

In the System Development Life Cycle first stage is System Study. It provides complete view of real physical system. It contains two phase; the first phase is preliminary survey and the second phase contains in depth and more detailed of project. System analyst prepared the system proposal after completion of system study. In light of the user’s requirements, proposed system contains findings, recommendations and solutions of limitations. System study undergoes through background analysis, findings or inferences etc.

Feasibility study: The main aim of the feasibility study is to find out exactly whether the developing product is technically and financially feasible. It also involves problem analysis and collection of relevant data which would be the foundation of the system. It is just like an abstract definition of the problem and formulation of the different solution strategies.

- The feasibility study finds the various solutions for the problem with reference to technical and financial parameter. Operational, Technical and economical are the areas in feasibility study[38].

Requirement Analysis and Specifications: The objective of this phase is to understand exact requirement of the customer. Careful analysis on requirement and document it properly. Project objectives should be clear in this phase. The goal of this phase is to collect all data and information in perspective understating the customer requirement. User requirements should be properly organized and documented in Software Requirement Specification (SRS) document.

Project planning

Project planning gets very importance during the project work. The feasibility study necessary during the process of planning. Concerned people should undertake the project if feasibility study is found satisfactory.

2.8.1 Components of a Software Project Management Plan

A software project management plan has three components: the work to be done, the resources with which to do it, and the money to pay for it all.

There are many of drawing up a project management plan. One of the best is IEEE Standard 1058[1998]. The components of the plan are shown in figure 6.
The IEEE project management plan is designed for use with all types of software products. It does not impose a specific life-cycle model or prescribe a specific methodology. The plan essentially is a framework, the contents of which tailored by each organization for a particular domain, development team, or technique [77].

Figure 6: The IEEE Project Management Plan framework

1. Overview
2. Materials Reference
3. Acronyms and Definitions
4. Organization of Project
   1. Plans of Managerial Process
   2. Plans of Technical Process
   3. Plans of Supporting Process
5. Additional plan

**Organization of SRS**

The Institute of Electrical and Electronics Engineers (IEEE) has published guidelines and standard to organize an SRS document [IEEE87, IEEE94]. Various kinds of project need their own requirements and no common methods for all. A guideline of organization of SRS is mentioned below. The initial two activities in the SRS are identical. Organization of an SRS is mentioned in diagram 7.0[47].

1. Introduction
2. The Overall Description
3. Specific Requirements
4. Change management Process
5. Document Approval
6. Supporting Information
2.8.2 Software Metrics

Software Engineering (SE) is a stable, quantitative engineering discipline. Its stability arises from the wide range of metrics evolved by software engineers to measure various aspects of the software. The advantage of metrics is that you can measure in quantitative terms the different aspects of software that need evaluation on an ongoing basis for estimation. It then helps to evolve standards for software development. A Good software metrics is the one that is simple to learn, easy to understand, has to clear objective and most important, is independent of technology, architecture or programming languages. Professional organizations build metrics database for process, project (product) and quality. These metrics from the basis for resource estimation, cost estimation and productivity computations [82].

Measurement is fundamental concept to any engineering discipline. Computer software needs measurement. Metrics of software can be used for measurement. The main aim is used to improve process on continuous basis during development. Software developers controls and evaluate the project by using the concept of measurement [47]. Measurement provides the action on path of during carried out the software development project process.

Establishing a Software Metrics Program has developed a comprehensive guidebook [PAR96] for establishing a “goal-driven” software metrics program. The guidebook suggest the following steps

1. Business goals
2. To learn as per requirement
3. Sub goals
4. Entities and attributes of sub goals
5. Measurement goals
6. Quantifiable questions
7. Data elements [69].

Function Point

Alan Albret while working for IBM, recognized the problem in size measurement in the 1970s, and developed a technique (which he called function point Analysis) which appeared to be a solution to the size measurement problem.[ALBR79, ALBR83]. With
reference to user acceptance level it calculate functionality depend on the request from user and in return. Therefore, it deals with the functionality being delivered, and not with the lines of code, source modules, files etc. Measuring size in this way has the advantage that the size measure is independent of the technology use to deliver the function. In other words, two identical counting systems, one written in 4GL and other in assembler would have the same function count. This makes sense to the user, because the object is to buy an accounting system, not lines of assemblers and it makes sense to the IT department, because they can measure the performances differences between the assembler and 4GL environments [STEP 95]. The principle of Albrecht’s function point analysis (FPA) is that a system is decomposed into functional units.

- Inputs
- Outputs
- Enquiries
- Internal logical files
- External interface files [47].

Several software metrics that have been developed to measure different aspects of software. Indirect measures are functionality, quality, complexity, efficiency, dependability, ease of use etc.
Design: The main objective of this phase is to transform the requirement specification into a structure which can be suitable for computer programming language. It provides the software architecture from the Software Requirement Specification document. Various types of tools like Data Flow Diagram (DFD), data dictionary, data definition, context diagrams, structure chart etc. are used. Design is an iterative process and translating into a blueprint for constructing the software [29]. Following (Table 02) is the different type of design process available and they have its own features. Table 02

<table>
<thead>
<tr>
<th>Data design</th>
<th>Make a thorough knowledge model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Design</td>
<td>Association of relevant parameters</td>
</tr>
<tr>
<td>Interface Design</td>
<td>Interrelation and transmit for the user</td>
</tr>
<tr>
<td>Procedural design</td>
<td>Transform framework elements into a procedural description</td>
</tr>
</tbody>
</table>

Good design has the following features

i. As per the analysis model it contains explicit requirement

ii. As per the customer desired, accommodate all implicit requirement

iii. It must be helpful to the code generator

iv. It reflect the complete view of the software [10].

Data Dictionary

In Software Engineering techniques, tools are used to reduce the complexity for development of software product. Different types of entity are used in the software product preparation process. To provide the information of relevant name, description and composite object name that includes under the title of Data dictionary in system model. It also includes representation of the entity, information about creator and date of creation. The main advantages of data dictionary are performed for name management mechanism and organizational information [75].
Data Flow Diagram

Data Flow Diagram used for top-level internal and external design specifications. It is a helpful mechanism in requirement analysis during communicating with the customer [68].

Structure Charts

⇒ In the architectural design structure chart is used. Hierarchical structure documentation and interconnections of the system displayed in the chart.

HIPO Diagrams

⇒ HIPO Diagrams (Hierarchy-Process-Input-Output) were developed at IBM. It represented scheme for top-down software development. It is helpful as external documentation for releasing the products [81].

Data flow analysis (DFA) is a technique with high cohesion for developing design modules. It is related with concept of analysis techniques. Here, DFA is presented in conjunction with structured system analysis. The input to the technique is a data flow diagram; a key point is that, once the DFD has been completed, the software designer has precise and complete information regarding the input to and output from the product.

Consider the flow of data in the product is represented by the DFD of figure 9-a. The product somehow transforms input into output. At some point in the DFD, the input ceases to be input and becomes some sort of internal data. Then, at some further point, these internal data take on the quality of output. This is shown in more detail in figure 9-b. The point at which the input loses the quality of being input and simply becomes internal data operated on by the product is termed the point of highest abstraction of input. The point of highest abstraction of output is similarly the first point in the flow of data at which the output can be identified as such, rather than as some sort of internal data [77].

![Data Flow Diagram](image1)

Figure 8-a: A data flow diagram showing flow of data and operations of product

![Data Flow Diagram](image2)

Figure 8-b: Points of highest abstraction of input and output
**Coding:** This phase is also called as implementation phase. The main aim of this phase is software design translate in to source code. Each component of design phase is implemented as a program module. Separately tested of program module is treated as unit testing. The following are the useful compiler phases for object code generation.

Lexical Analysis, initially find out the identifiers after analyzing the program, replaces it with an internal representation. Mentioned the description for building the symbol table. In Syntax analysis, output comes from lexical analysis for constructing the syntax tree with correct description of real code. Object code produces through code generation. It takes various steps for successful completion [37].

![Flowchart Diagram](image)

*Figure 09:F further modularization of the code-generation module*

**Integration & System testing:** In this phase, those modules created in coding phase, should be integrated in planned manner. It includes step by step procedure for integration.
of modules and tests it stepwise till completion of the system. The system testing should be carried out according to its requirement as specified in the SRS document.

**Terms of Software Failure**

Software testing is one of the important aspects before to launch the product for actual working. Various terms are used while testing the software and result. Defects, failure and fault that create the errors while testing. [70]. Problem, error, and bug are probably the most generic terms used.

In the past many software projects have suffered because designers started implementing something without determining whether they were building what the customer exactly wanted. Software engineers systematically organised the essence of all past programming experience and creates various types techniques, methodology and discipline.

**Tests**

Tests are required before to launch the product in the market. Design the test process that should protect the bugs. Results are examined after supplying the proper input. All type of testing activities incorporated in the formal procedure of testing.

**2.8.3 Testing and Levels**

Different testing processes for the software system are available. They are Unit, Component, Integration and System testing. The main intention of each testing is different.

**Unit testing** - In software engineering methodology used Object Oriented Programming technique. The system describes in different modules. To test it separately, assembled or compiled linked, and loaded a part of the software. When the test done, if fault occurs that indicates a unit bug

**Integration Testing** - Component contain one or more part of the software system. Further it may join one or more component for Integration Testing. It is high level component testing. The objective behind the integration testing is to exhibit the problems that arise from the aggregated components [8].

**Maintenance**: Technology is changing rapidly. Due to changing technology and new requirements of the customer, it requires to modify the existing software product so
that product should have utility for long duration. In this phase software engineers have
the facility to modify the software product after delivered to the customer.

Software Development Problems

Two principal software development problems have been identified (Brooks, 1987)

1. Conceptual problem. Specifying, designing, testing the conceptual construct
underlying a software system.

2. Representation problem. Representing software and testing the fidelity of a
representation

The conceptual problem is considered hard because the essence of a software entity is
a construct of interlocking concepts. These concepts can be found in the data sets,
relationships among data items, algorithms, and invocations of functions within a
program. By contrast, the representation problem is considered easier because it
concerns accidental features of software. The distinction between the accidental and
essential features of an object was introduced by Aristotle in a book called Topics
written about 340 B.C. (Barnes, 1984). A feature of a thing is considered accidental if
the thing can persist with or without the feature [43].

2.8.4 SE Data Collection: Role of Validation

- SE data is error prone and needs careful validation.
- Validation should be performed concurrently with software development and
  Data collection.
- Automated tools can be useful in data collection and validation, e.g. range
  checks, conditional entries, check digit, etc.
- Training, clear guidance, instruction, and education about the use of data, etc. can
  help in improving the quality of data.
- Where possible, the data collection activity should be merged with the
  configuration management activity [71].

2.8.5 Software Quality Assurance (SQA)

Quality is achieved by building it into the software all throughout the project
lifecycle; it cannot be added at the end of software development. This implies that the
relationship between software quality and software development process is known that,
by proper engineering of this process, a product of the appropriate quality can be
produced. The quality of the software as it is developed must be continually examined and satisfactory level should be there. This evaluation must be planned and documented and made known to those who execute the software development project. The planning of how quality is to be built in and how it is to be evaluated is recorded in quality assurance plan.

2.8.6 Contents of the Software Quality Assurance Plan

As per the IEEE principle for software Quality Assurance Plans states that the plan should contain the below defined sections.

- **Purpose**
  This states that the specific purpose and scope of the SQA plan. It names the software products that it covers and describes the use to which they will be put.

- **Reference documents**
  A complete list of the documents referenced in the plan

- **Management**
  IEEE standard lays down three aspects that should be covered in this section of the Quality Assurance Plan; Organization, task and responsibilities.

- **Documentation**
  The basic purpose of this section of the Software Quality Assurance plan is to describe the documentation to be produced and how it is to be reviewed. All the documentation relating to the development, verification, use and maintenance of the software will be listed.

- **Standards, Practices and Conventions**
  This section of the Software Quality Assurance plan should contain at a minimum, the following:
  - Documentation standards
  - Logic structure standards
  - Coding standards
  - Commentary standards

- **Review and Audits**
  This section of the Software Quality Assurance plan will state which technical and managerial reviews will be undertaken and how they will be carried out.
Project can separately keep a schedule of provisional dates for conducting these reviews.

**Problem Reporting and Corrective Action**

This section of the Software Quality Assurance plan will describe the system, which ensures that software problems are documented and resolved. It should be a closed-loop system. All the problems should be promptly reported / escalated at appropriate level, acted upon and resolved. Each problem should be capable of being tracked throughout the life cycle of the software project. Each problem should be analyzed to determine its significance and causes and classified by category (such as requirements, design, coding etc.) and each problem must have a severity level and priority number (to enable action on it). Trends in the problems reported should be identified.

**Tools, Techniques and Methodologies**

This section of the SQA plan should identify the special software tools, techniques and methodologies employed that support quality assurance, state their purposes and describe their uses.

**Code Control**

In a software project, this is likely to be implemented in conjunction with the library function. The library receives and maintains copies of all software tools and documentation. The library will issue all material and ensure that the most recently Authorized Version is the one routinely available. Access to code files is controlled to ensure that no unauthorized use or modification takes place. The library will ensure that the correct version of software is submitted for testing.

**Media Control and Backup**

The section of the Software Quality Assurance Plan will describe how the media are to be protected from unauthorized access or damage. One would expect to see details of storage arrangements that protect the media from harmful environmental conditions. Actually, Disaster Recovery Plan is a much larger aspect of Media Control and backup. Security threats to a software project come from the following environmental factors:

- Fire Damage
- Water Damage
- Energy Variations
- Structural Damage
- Pollution
- Unauthorized Intrusion
- Viruses and Worms
- Misuse of Software, Data and Services

**Supplier Control**

This has relevance while outsourcing some components of a software project. It is important that externally developed software is of the appropriate quality. From the CMM-I angle, its importance comes from the fact that “Supplier Agreement Management” (SAM) is an all new process area under CMM-I.

**Records Collection, maintenance and Retention**

Any successful project will undergo substantial maintenance over a long period. As such, it is important to ensure that all the documentation necessary to undertake this quickly, efficiently and cheaply is going to be available when required. This documentation must be produced during development. It will be retained. This section of the Software Quality Assurance Plan will identify who is responsible for ensuring that the appropriate documentation produced and stored, and under what conditions [62].

**2.9 Components of Software Engineering**

It has two type components first is, system engineering approach and second is, development engineering approach. The features of the system based on design, quality and scope. Initially, understand the system by the software engineer. The system includes different areas like business system. Computer software used in the business system. Software Engineering Methodology (SEM). The SEM steps are as under

- Aim of the system
- Limits of the system
- Intelligence of the system
- Association between various components
- Overall connections in processes
- Role of hardware and software.
• Operational and functional requirements
• Model the software
• Discuss the system with concern people

The second component of SE is development engineering. In this methodology concentrates on targets for shifting the system needs as a goal and follow the series of steps to achieve it.

The development engineering steps are –

• Need of definition and details
• Preliminary plan for solutions
• Design & construction for delivery
• Planning for product
• Testing components
• Combine testing
• Implementation plan
• Execution
• Modification
• Maintenance

2.10 Software Engineering Tools

Software engineering uses some basic engineering tools that aid engineering development just like any other engineering discipline. These tools are evolved over a period and have become an industry standard because they show great promise as utility tools in the industry. Software engineers are used these tools for development of software product. The product may be small or big simple or complex product or custom software. They are useful at all time in all domains. The application of these tools assures to a great extent the smooth and efficient development of software. These tools to control effort, cost and associated risk in software development.

Following are the software engineering tools

• Modeling
• Analysis
• Requirement engineering
• Work-Breakdown Structure (WBS)
• WBS scheduling
• Prototyping
• CASE

Modeling

A model is an abstract representation of scenario or situation built to understand and analyze it. Since it is an abstraction, it considers those aspects and factors of the situation that are prominent, dominant and critical. Modeling helps to express complex ideas, improves the learning process and helps manipulation and communication between development teams.

Analysis

A good analysis helps to understand situations, brings clarity in cause and effect relationships and reduces ambiguity. It further helps to priorities various critical aspects to be built in the model and prototype. Analysis provides multiple views of the situation, providing better insight into the scenario or situation

Requirement Engineering

Requirement Engineering is a process of ascertaining the customer requirements described in terms of environment, goals, problems and expectations and solutions. RE further puts these details on ascertainment and confirmation by all (customer, end users and stakeholders) in to well-defined, structured document known as Requirement Definition and Description (RDD). Once the RDD is finalized it is used to determine the Software Requirement Specification (SRS).

Work Breakdown Structuring (WBS)

The ‘work’ here is a larger or bigger entity broken down into components or modules. These in turn are broken further into tasks, tasks into activities and activity into process.
Good WBS covers the entire scope and ensures that when it is completed, the desired goal of the software is achieved. Further WBS helps to plan and control the software development effectively.

**WBS scheduling**

Once the WBS is finalized it needs to be scheduled on a time scale to ensure that WBS units ‘start on time and finish on time’ and complete the overall software delivery as promised to the customer.

**Prototype**

Prototype is a representative model of an entity, which you want to achieve successfully. The prototype could be of software, process or product made essentially to understand the requirement. The prototyping process begins with listening to customer needs, problems and solutions proposed. The software engineer has to identify the areas that need confirmation and testing. Then those needs that are critical to customer goals should be identified and alternative solutions found to meet those needs.
CASE- Computer Aided Software Engineering

CASE is a tool to automate a certain processes in the software development cycle. Since automation is the methodology, it not only expedites the development but provides a comprehensive approach to all aspects of software development. A battery of tools put together is called a CASE tool. When these tools are integrated as set, the set provides an automated CASE tool termed as I-CASE (Integrated CASE) [82].

Agile Principles

- The top priority should be given to customer’s satisfaction. To supply fast and consistently handover the quality software product
- Accept the modification in requirements
- In short time scales deliver software product
- Work together with developers and client
- Continuous attention to technical excellence [6].

2.11 The Capability Maturity Model Integration (CMMI)

For maintaining the quality of software product Software Engineering Institute (SEI) developed the CMM model. The model helps to improve the process and steps for development of software product. The Capability Maturity Model (CMM) mentioned general belief for software development. Mostly it is based on experience, abstract and general framework. It is a reference model for the people who are working in the software development Process [18].

Risk is handled by different people in different ways based on their preferences and attitude towards risk. People differ in their tolerance to risk. The difference occurs due to the different utility functions on which exposure is measured.

2.12 Nature of Risk

Table 3 gives broadly the types of risk that are common in software development scenarios. The nature of risk and its class are, however, fixed, and its class are, however, fixed, and they are stated below:
Table 3

- **Technology**: These risks fall in the domain of hardware, software and network communication technology, which are subject to continuous improvement and may lead to obsolescence.

- **People Risks**: These risks are associated with persons in the teams and risk concerns availability, capability, skills, maturity and so on.

- **Organizational Risk**: These are in the area of development organization and customer organization. The environment in both the organizations is a matter of concern.

- **Requirement Risks**: This is about clarity, completeness and confirmation of the requirement by the end users and customer. It is about the volatility of the requirement.

- **Estimation Risks**: Not being able to judge in precise terms the components of the software, reusability and system characteristics affecting the size, effort and schedule.

**Software Risk Management (SRM)**

Under the five steps Software Risk Management (SRM) process is built.

Identify, Analyze, Plan, Track, Resolve are the steps. The process is very active in nature and controlled systematically during the development of life cycle. Figure 11 define the process model of SRM.

![SRM Process Model](image-url)
Identify Risk

Risk identification is a series of actions which provides detail relevant information to minimize risk.

Analyze and Assess Risk

To estimate and evaluation of risk for development process in different activity Risk Analysis methodology is useful.

Plan Risk

After identified the risk, based on earlier list of risk. The risk plan take action to handle it properly which analyzes in the initially. RAP determine the various strategies which is mentioned in the list.

Resolve Risk

The process of Risk resolution techniques depend on various tools, methodology and database on which the risk action plan based. The main intention behind the process to minimize the risk as required in acceptable level. The result of this process is to reduced rework and actions

Risk Mitigation Through RMMM Plan

Risk Mitigation Planning

Risk Mitigation or reducing the impact of risk is done through an RMMM plan. An RMMM plan deals with mitigation, monitoring and management of risks in a systematic manner. The steps in an RMMM plan are as under.

- Prepare a prioritized risk list based on risk exposure and its severity index
- Determine risk resolution strategy for each
- Design an action plan based on resolution strategy to deal with risk
- Institute a monitoring plan through systematic review
- Take corrective measures to control the impacts

The most of the work of human being is done by the software. It is made by the software engineers. The engineers have a wide knowledge to solve the problem efficiently and resolve the complex problem by using software methodology techniques. Software carries out most of the significant task with their appropriate goal. The main
The aim of software engineering is to create processes and techniques for the good production of software products that will be reengineered as and when required. It should be possible to determine in advance about the time and effort required to produce the final product. Software Development activities follow a process while developing a software product. A key component of any software development process is the life cycle model on which the process is based. Life cycle of the software process starts from exploration of concepts and ends at the retirement of the software. Software Development Life Cycle (SDLC) is nothing but a process of creating and maintaining a good software product. It is a series of identifiable stages that a software product undergoes during its life time. Several different activities are carried out in each life cycle phase. The significant objective of SDLC is to deliver an application that fulfills the customer requirements. This can be achieved through a systematic, step-wise, and cost-effective way called the software development model. Generally, the software development process begins with feasibility study of the project. If the project is technically and financially feasible, then study on second phase i.e., project requirement analysis and specification. Then to start design, coding, testing phases. These phases referred as phase of life cycle. During these phases one important stage is Software Requirement Specification (SRS) phase then next process is to estimate size, cost and development time of the project [82].

### 2.13 Reengineering

Reengineering is a process of evaluating and changing the system and makes it to be recently created system. Reengineering concept provides the two way process to satisfy the requirement. First is the forward engineering and second is the reverse engineering (Figure 12). In forward engineering focus on various stages which were earlier implemented for building the software product. Structure, coding, implementation etc. these are the some important components in forward engineering. In reverse engineering concentrates in working system. Identification, analysis and interrelationship are the main areas in reverse engineering [43].
Figure 12 Reengineering model