CHAPTER – 1
INTRODUCTION

1.1 Software Development Life Cycle (SDLC)

SDLC, Software Development Life Cycle is a process followed by software industries to design, develop and test high quality software's. The SDLC aims to produce quality software that meets customer expectations, reaches completion within times and within budget constraints.

SDLC is a framework which explains various tasks at each step in the software development process. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that depicts all the tasks required for development and maintenance of software.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure 1.1 is a graphical illustration of the different phases of generic software development life cycle

![Figure 1.1 Different Phases of SDLC](image-url)
1.1.1 **A typical Software Development life cycle consists of the following stages:**

**Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most vital and fundamental stage in SDLC. It is performed by the experienced members of the team with clients input, department of sales, and surveys related to marketing and consulting with domain experts in the industry. This information is then taken forward for planning the basic project approach and then conduct product feasibility study in the operational, economical, and technical areas. Planning for the quality and risks associated with the project is also carried out in the planning stage. The result of the technical feasibility report is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

**Stage 2: Defining Requirements**

Further clearly defining and documenting the product requirements and get them approved from the customer or the market analysts is carried out. This is done through Software Requirement Specification (SRS) document describing complete product requirements to be designed and developed during the project life cycle.

**Stage 3: Designing the product architecture**

SRS is considered as a point of reference for product architects to conclude the best architecture for the product under development stage. Based on the requirements specified in SRS, the product architecture is proposed and documented in a DDS - Design Document Specification.

A design approach specifies all the architectural modules of the product along with its communication and data flow representation with other modules. The internal design of all the modules should be clearly stated with the sense of completeness in DDS.

**Stage 4: Building or Developing the Product**

In this stage, the practical development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much delays.
Stage 5: Testing the Product
This stage refers to the testing stage where functional and non-functional requirements of the product are satisfied are checked, reports of products defects are generated, tracked, and retested, until the developed product meets the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance
After testing is done and product is to be deployed it is released formally in the target market. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing) where we can identify some real time errors if any. Then based on the feedback, the product may be released as it is or with suggested enhancements in the market. After the product is released in the market, its maintenance is carried out for the customers.

1.2 Small and medium Undertakings
Small programming firms normally work in profoundly changing situations while being obliged by restricted and firmly planned assets, predominately settled expenses of improvement, and high reliance on couple of vast players inside of the business. By small & medium level firms we mean firms with under 100 representatives.
As per the procurement of Micro, Small and Medium Enterprises Development (MSMED) Act, 2006 the Micro, Small and Medium Enterprises (MSME) are ordered in two Classes:
(a) Manufacturing Enterprises : The ventures occupied with the assembling or generation of merchandise relating to any industry determined in the first timetable to the businesses (Development and regulation) Act, 1951) or utilizing plant and hardware during the time spent worth expansion to the last item having an unmistakable name or character or utilize. The Manufacturing Enterprise are characterized as far as interest in Plant and Machinery.
(b) Service Enterprises: The undertakings occupied with giving or rendering of administrations and are characterized as far as interest in gear.
Table 1.1 Classification of Micro, Small & Medium Enterprises

<table>
<thead>
<tr>
<th>Manufacturing Sector</th>
<th>Investment in plant &amp; machinery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprises</td>
<td></td>
</tr>
<tr>
<td>Micro Enterprises</td>
<td>Does not exceed twenty five lakh rupees</td>
</tr>
<tr>
<td>Small Enterprises</td>
<td>More than twenty five lakh rupees but does not exceed five crore rupees</td>
</tr>
<tr>
<td>Medium Enterprises</td>
<td>More than five crore rupees but does not exceed ten crore rupees</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service Sector</th>
<th>Investment in equipment’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprises</td>
<td></td>
</tr>
<tr>
<td>Micro Enterprises</td>
<td>Does not exceed ten lakh rupees:</td>
</tr>
<tr>
<td>Small Enterprises</td>
<td>More than ten lakh rupees but does not exceed two crore rupees</td>
</tr>
<tr>
<td>Medium Enterprises</td>
<td>More than two crore rupees but does not exceed five crore rupees</td>
</tr>
</tbody>
</table>

1.3 Lean Software Development

Lean development focuses on creating high-value products by removing waste from the development cycle. Lean software development is a software development model adapted from lean manufacturing and agile development principles. The Lean model focuses on client criticism and waste lessening and also various other helpful rule that actualize the fundamentals of Lean improvement in a product situation. Lean programming improvement advances a versatile, incremental model that makes a quality item speedier and more typically than other improvement forms.

1.3.1 Principles of Lean Software Development

In Implementing Lean Software Development, Mary and Tom Poppendieck show how the seven principles of lean manufacturing can be applied to optimize the whole IT value stream. These principles are:

**Eliminate waste:** Lean deduction backers respect any movement that does not specifically increase the value of the completed item as waste. The three greatest
wellsprings of waste in programming improvement are the expansion of unrequired components, undertaking agitate and intersection authoritative limits (especially in the middle of partners and advancement groups). To diminish waste it is important that advancement groups be permitted to self-compose and work in a way that mirrors the work they're attempting to fulfill. Walker Royce contends in "Enhancing Software Economics" that the essential advantage of present day iterative/dexterous strategies is the lessening of scrap and revamp late in the lifecycle.

**Build in quality:** Your procedure ought not permit imperfections to happen in any case, however when this isn't conceivable you ought to work in a manner that you do a touch of work, accept it, settle any issues that you find, and after that emphasize. Examining sometime later, and lining up imperfections to be settled eventually, isn't as compelling. Light-footed practices which incorporate quality with your procedure incorporate test driven improvement (TDD) and non-solo advancement practices, for example, pair programming and displaying with others.

**Create knowledge:** Planning is helpful, yet learning is key. You need to advance systems, for example, iterative improvement, that help groups find what partners truly need and follow up on that information. It's additionally imperative for a group to consistently ponder what they're doing and after that demonstration to enhance their methodology.

**Defer commitment:** It's not important to begin defining so as to programme improvement a complete determination, and actually that gives off an impression of being a faulty procedure, best case scenario. You can bolster the business adequately through adaptable architectures that are change tolerant and by planning irreversible choices to the last conceivable minute. Every now and then, conceding responsibility requires the capacity to nearly couple end-to-end business situations to abilities created in various applications by different undertakings.

**Deliver quickly:** It is conceivable to convey top notch frameworks rapidly. By restricting the work of a group to its capability, which is reflected by the group's speed (this is the quantity of "focuses" of usefulness which a group conveys every emphasis), one can set up a dependable and repeatable stream of work. A viable association doesn't request groups accomplish more than they are prepared to do, however rather asks them to self-
compose and figure out what they can fulfill. Obliging these groups to conveying conceivably shippable arrangements all the time rouses them to stay concentrated on persistently including quality

**Respect people:** The Poppendieck additionally watch that feasible point of preference is picked up from connected with, deduction individuals. The suggestion is that one require an incline administration technique that spotlights on spurring and empowering IT groups—not on controlling them

**Optimize the whole:** On the off chance that you need to be powerful at an answer you must take a gander at the master plan. One has to comprehend the abnormal state business forms that individual ventures bolster—forms that regularly cross various frameworks. You have to oversee projects of interrelated frameworks so you can convey a complete item to your partners. Estimations ought to address how well you're conveying business esteem, in light of the fact that that is the sole purpose behind your IT department

**1.4 Lean thinking is important for scaling agile in several ways:**

1. Lean gives a clarification to why a large portion of the light-footed practices work. For instance, Agile Modeling's practices of light weight, introductory necessities imagining took after by cycle demonstrating and in the Just In Time (JIT) model raging work on the grounds that they reflect delay of duty with respect to what should be fabricated until it's really required, and the practices dispose of waste on the grounds that you're just displaying what should be manufactured.

2. Lean offers knowledge into systems for enhancing your product process. For instance, by comprehension the wellspring of waste in IT you can start to distinguish it and afterward kill it.

3. Lean standards give a philosophical establishment to scaling dexterous methodologies.

4. It gives procedures to recognizing waste. Value stream mapping, a procedure regular inside of the incline group whereby you show a procedure and after that distinguish the amount of time is spent on quality included work versus hold up time, figures general time proficiency of what you're doing. Quality stream maps are a direct approach to enlighten one’s IT procedures, giving knowledge into where huge issues exist.
1.5 Capability Maturity Model (CMM)

It is a bench-mark used as a measurement tool to check the maturity of an organization’s all round software process. It is a methodology adopted by software firms to develop and refine its development process. CMM can be used to assess an organization against a scale of five levels as depicted in figure 1.2

Exhibited characteristics of each maturity level:

Maturity Level 1 – Initial: This level can be generally seen in small companies, here there are absence of standard procedures for developing a software. It also lacks project-tracking system which can predict time constraints and track all activities taking place.

Stable environment is not provided by the company, the only factor responsible for the success of the company is its human resource, and they completely rely on people to carry out the project on their own talents.
The products and services delivered by such organizations work but there are no standard procedures adopted by the company for the development. These organizations sometimes promise more but deliver less and fail to repeat their success stories consistently.

Maturity Level 2 – Managed: This level can be generally observed in small and mid-sized firms. Company management controls and procedures, but lacks consistency or team work among various groups.

Achievement of the organization are it manages to achieve its goals. In other words, the projects of the organization have ensured that requirements of the clients are met and that processes adopted are well planned, duly performed, quantitatively measured, and controlled.

The process executed with success are retained at tough times so that they can be reused, performing of projects and its management are very well documented.

Maturity Level 3 – Defined: This Level can be generally observed in large organizations. Company has its standard set of processes and controls the entire organization, developers can switch between projects more easily and customers get feedback consistently from different groups.

Maturity Level 4 – Quantitatively Managed: This level can be seen in Multinational companies. In addition to implementing standard processes, company has its own standard processes to measure the quality of those processes across all projects. Quality objectives of the company are very well defined and established and are used in managing procedures, quality performance and total quality management are very well understood in terms of statistics and is delivered consistently throughout the lifecycle of the product. Performance measurers and quality are adopted to support decision making at all levels in the company.
Maturity Level 5 – Optimizing: This level can be observed in companies which have proved themselves since decades and are ruling the market with their own procedures and strategies in the whole world.

The company’s focus is to continuously improve its performance and satisfies the customer, the company rapidly responds to all situations both within and outside environment.

Optimizes all its processes and standards adopted.

1.6 Failure Mode Effective Analysis

FMEA is a philosophy to discover potential failures before they happen. While it distinguishes singular failure modes, its essential advantage is the early ID of framework failure modes so an answer can be intended to alleviate the potential failure. It is a philosophy to plan unwavering quality into a framework. In a FMEA, numerical weights can be connected to the probabilities of every failure, and also the seriousness of the outcomes. FMEA is an exceptionally practical, simple to learn, and gainful approach to outline a more solid framework. In spite of the fact that this strategy was not initially made for programming frameworks, we can make an interpretation of the standards over to programming and exploit the numerous advantages that FMEA brings to the table.

These advantages include:

• Facilitates early recognizable proof of failure focuses and framework interface issues
• Yields a superior comprehension of arranging/booking by uncovering extra work endeavors
• Enables early test arranging
• Provides a solitary worksheet rundown of examination
• Requires negligible preparing to take part, or even lead a FMEA
• Increases unwavering quality in items, the result of this procedure is a solitary FMEA worksheet containing a rundown of failure modes organized by danger in the framework under study. This organized rundown recognizes the most important spots to concentrate additional plan and test endeavors to deliver a more dependable system
1.6.1 Key Elements of FMEA

A run of the mill FMEA is a group action, finished in one or more gatherings. The proprietor of the framework investigated, venture leads, QA, and no less than one space master normally go to the meeting(s). Anyone with even direct information of the framework can likewise go to and be of incredible advantage to the result. Amid the meeting, these seven crucial steps give direction through the procedure:

1. Characterize Failure Modes – What can turn out badly here?
2. Characterize Effects – What will happen then?
3. Portray Targets – Who will experience the ill effects of the failure?
4. Discover Root Causes – Why will that happen?
5. Organize the Risks – What is the probability?
6. Characterize Solution Actions – How can this be anticipated?
7. Characterize Current Prevention and Detection Methods – What is as of now being finished? These strides are rehashed all through the meeting, bringing about the FMEA worksheet with an organized rundown of dangers and their related failures in the framework.

To perform a FMEA in a product framework, the Functional FMEA model is utilized. The fundamental distinction from a Process FMEA is that for the Software FMEA there is no pre-characterized, straight stream that can be replicated from a work guideline. In the case in Section 3, the work directions are the beginning stage for the FMEA steps. In a product framework choices are set aside a few minutes to decide the proper way. This convolutes the meaning of the strides expected to start the FMEA worksheet. Another major contrast in the middle of Process and Software FMEA is that, for programming, it is not recommended to cover 100% of the configuration. Rather, center endeavors where failures are in all likelihood or where the results are extreme. These ought to be constrained to a specific sub-framework or even another element of a framework. With a specific end goal to characterize steps, first separate the framework into Function Blocks to make a Functional Block Diagram (FBD). A capacity square is an abnormal state portrayal of a bit of the product framework. The level of point of interest that goes into the capacity squares is a huge choice and can be impacted by an assortment of variables, for example,
• New advancements or equipment present – The group might need to invest more energy in regions where new advances or equipment are available, so more definite pieces ought to portray these regions.

• Overall certainty of the subsystem – Was there any re-utilized code as a part of the framework? These parts can be depicted with bigger, less illustrative squares, so more vitality is engaged where issues are more probable.

• Safety concerns – When wellbeing is a worry, useful squares ought to be more point by point so there is less risk for an oversight or supposition.

The most advantage of a Software FMEA will be acknowledged in right on time periods of configuration, in a perfect world before any code has been composed yet after a larger part of the necessities are characterized. At the point when performed early, the FMEA can uncover feeble purposes of the framework. Arrangements can be planned in equipment, programming, or both that can keep away from expensive outline changes in later venture stages. Having the adaptability to decide the most cost effective and most dependable arrangement within the near future is the biggest advantage of performing the FMEA. Test advancement and arranging can likewise be helped when performed ahead of schedule by knowing extra testing endeavors in advance. These additional endeavors may originate from:

• Verification required for all executed arrangement activities

• Test cases can be made where the "Present Prevention/Detection Methods" segment is void.

• More testing endeavors can be coordinated towards high RPN values. While the most advantages of doing a Software FMEA are acknowledged in the soonest plan stages, numerous advantages can even now be had in the later stages also. The least secure segments of the configuration will in any case be recognized, and numerous arrangements can in any case be produced. In any case, this ordinarily includes some significant downfalls of calendar or spending plan, and the adaptability to tackle issues in equipment may be lost.

FMEA is a broadly utilized and acknowledged technique for unwavering quality building. Its motivation is to distinguish conceivable failures, assess their impact on the
framework, and propose answers for alleviate these impacts. FMEA is most ordinarily utilized on the procedure and equipment levels, yet is turning out to be more ordinary in the product business. A few advantages of utilizing FMEA on a product framework include:

- Early distinguishing proof of single failure focuses and framework interface issues
- Getting a superior thought for arranging/booking
- Assisting in ahead of schedule test arranging
- Catching issues sufficiently early to be tackled in equipment of programming
- Single worksheet summary of the analysis