Introduction to Software Engineering

What is Legacy Software?

- **A legacy system is outdated computing software and/or hardware that is still in use.** The system still meets the needs it was originally designed for, but doesn't allow for growth. What a legacy system does now for the company is all it will ever do. A legacy system's older technology won't allow it to interact with newer systems.

The reasons are varied as to why a company would continue to use a legacy system.

- **Investment**: Although maintaining a legacy system is expensive over time, upgrading to a new system requires an up-front investment, both in dollars and manpower.
- **Fear:** Change is hard, and moving a whole company —or even a single department — to a new system can inspire some internal resistance.
- **Difficulty**: The legacy software may be built with an obsolete programming language that makes it hard to find personnel with the skills to make the migration. There may be little documentation about the system and the original developers have left the company. Sometimes simply planning the migration of data from a legacy system and defining the scope of requirements for a new system are overwhelming.

Changes needed in legacy software

1. The software must be adopted to meet the needs of new computing environments or technology.
2. The software must be enhanced to implement new business requirements.
3. The software must be extended to make it interoperable with more modern systems or databases.
4. The software must be re-architected to make it viable within a network environment.

Problems caused by legacy systems

1. **Maintenance is costly :**

   Maintenance is to expected with any system, but the cost of maintaining a legacy system is extensive. Maintenance keeps the legacy system running, but at the same time, the company is throwing good money after bad. The status quo is maintained, but there's never a chance for growth with the legacy system.

2. **New systems don't integrate**

   As a company matures, adding new systems is necessary to stay competitive in today's world. But the older technology of a legacy system may not be able to interact with a new system. A department still using a legacy system won't receive all the benefits that a new system offers.

3. **Security gets weaker by the day:**

A data breach can cost a company dearly, and legacy systems are more weak to hackers than newer systems. Legacy systems by definition have outdated data security measures, such as hard-coded passwords. That wasn't a problem when the system was built, but it is now.

4. **Compliance is much harder**

Organizations today must abide by strict sets of compliance regulations. As these regulations continue to evolve, a legacy system may not be equipped to meet them.

Compliance regulations like the GDPR, for example, require a company to know (and prove) what customer data they have, where it is, and who is accessing it. Companies with customer data need to maintain well-governed records, which is much harder (if not impossible) in outdated, siloed systems.

Legacy software modernization

There are several proven strategies to upgrade legacy systems:

Total Transformation. The entire system is rebuilt using new technology, and the old system is sunsetted. The new system is built from scratch using modern standard platforms or built using a third-party package as a foundation layer.

Gradual Replacement. A component/functional block of a legacy system is replaced with a new technology and moved to production as a separate application while the rest of the system continues to use old technology. With time, remaining components/functional blocks are replaced with separate applications and gradually the entire system is rebuilt.

Duct Tape Approach. Localized, small-scale changes are performed using new technology to address specific issues in the application, while the application core architecture and technology remain the same. A popular approach is to build a new application that will be bolted to the main application to bridge the gap in functionality.

Improve Existing. The existing legacy system is modernized to offer better results through improved design. Typically, the core technology stack remains the same or a few minor additions may be introduced.

No System Change. Clients are taking a wait-and-watch approach and not going for any modernization drive or systems change.

Software Engineering–Layered Technology

Software Engineering is a layered technology. It is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of systems.

The main objective of software engineering layers is to help software developers obtain high-quality software. There are four types of layers in Software Engineering such as – **Tools, methods, process, A quality focus**.

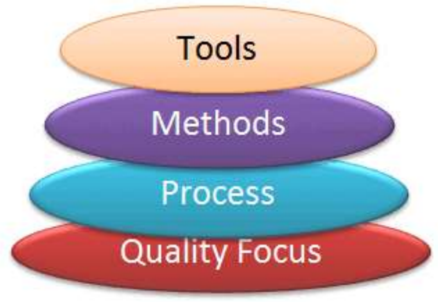

Fig: Software Engineering Layers

A quality Focus:

- Main principle of Software Engineering is Quality Focus.
- An engineering approach must have a focus on quality.
- It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.
- Total Quality Management (TQM), Six Sigma, ISO 9001, ISO 9000-3, CAPABILITY MATURITY MODEL (CMM), CMMI & similar approaches encourages a continuous process improvement culture.

Process:

- It is a foundation of Software Engineering
- It is the glue the holds the technology layers
- It defines a framework with activities for effective delivery of software engineering technology.
- The software process covers all the activities, actions, and tasks required to be carried out for software development.

Process activities are listed below:-

- **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
- **Planning:** It basically means drawing a map for reduced the complication of development.

- **Modeling:** In this process, a model is created according to the client for better understanding.
- **Construction:** It includes the coding and testing of the problem.
- **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.

Methods:

- The method provides the answers of all 'how-to' that are asked during the process.
- It provides the technical way to implement the software.
- It includes collection of tasks starting from communication, requirement analysis, analysis and design modelling, program construction, testing and support.

Tools

- The software engineering tool is an automated support for the software development.
- The tools are integrated i.e the information created by one tool can be used by the other tool.
- **For example:** The Microsoft publisher can be used as a web designing tool.