**Abstract**

In this report we will discuss different stages of SDLC in traditional way. The architecture will be studied and we will be building a connection between SDLC and architecture. The discussion will show why it is important to reorient SDLC with respect to architecture. The further discussion will reveal how SDLC incorporated architecture in modern day software engineering practices and how it is taken care of during different phases of SDLC.

**Introduction**

In software engineering, architecture has a prime importance and is extensively involved in engineering process throughout the globe. In the initial days of software development architecture was not that much famous. The traditional understanding of SDLC were mostly without discussion of architecture. But now in modern day software engineering, the process is carried out by keeping development challenges, future plans, and factors like reusability, scalability and maintenance. All these factors causes the strong interference of software architecture in the field of software engineering. Almost every part and phase of software development involves architecture in some direct or indirect way. Since the architecture is so much involved in software engineering process, we need to elaborate the phases of SDLC to see the involvement of architecture at different stages of software development.

**SDLC at a glance**

**Architecture and its importance**

Software architecture is the foundation of a software system. Like other types of engineering, the foundation has a profound effect on the quality of what is built on top of it. As such, it holds a great deal of importance in terms of the successful development, and eventual maintenance, of the system.

The software architecture of a system **depicts the system's organization or structure, and provides an explanation of how it behaves**. A system represents the collection of components that accomplish a specific function or set of functions. A software architect makes important decisions regarding the software that goes on to define its overall integrity. A good software architecture helps define attributes such as performance, quality, scalability, maintainability, manageability, and usability.

**Why Does Software Architecture Matter?**

An organized software architecture helps to ensure the longevity of the software’s internal quality.

Consider two similar products. Both are launched within a month-long gap and aims to add new features when they complete three months.

There are two scenarios:

* Product A launched in September 2022. This project supports a messy source code because the [development team](https://www.netsolutions.com/hire-developers) wanted to launch and monopolize the market as early as possible.
* Product B launched in November 2022. This project has a software architecture that is well-structured and organized. The development team works on the design and architectural decisions early in the process and prioritizes quality over faster launch.

**Which Product will be more successful: A or B?**

Product A might monopolize the market initially and convert better. However, product adoption will eventually subside because the messy code will lead to [technical debt](https://www.netsolutions.com/insights/what-is-technical-debt/) pileups. These pileups will, in turn, make it challenging to introduce new updates and bug fixes on the fly.

Product B might have a market entry gap, but it will be easier to maintain a faster shipping cadence. The customer needs will be looked after without breaking the shipping cadence, thus making for a larger win.

## ****Why is software architecture important?****

A proper foundation laid down by a software system's architecture yields a number of benefits. Let's take a deeper look at those benefits:

### **Defining a solution to meet requirements**

Software strives to meet all functional, non-functional, technical, and operational requirements. Working closely with stakeholders, such as domain experts, business analysts, product owners, and end users, allows requirements to be identified and understood. A software architecture defines a solution that will meet those requirements.

### **Easing communication among stakeholders**

Software architecture and its documentation allow you to communicate the software architecture and explain it to others. It can form the basis for discussions related to aspects of the project, such as costs and duration. A software architecture is abstract enough that many stakeholders, with little or no guidance, should be able to reason about the software system

### **Serves as training for team members**

The system's architecture and its documentation serve as training for the developers on the team. By learning the various structures and elements of the system, and how they are supposed to interact, they learn the proper way in which the functionality is to be implemented. A software development team may experience change, such as having new team members join or existing ones leave. The introduction and orientation of new members to a team often takes time. A well-thought-out architecture can make it easier for developers to transition to the team.

### **Managing change**

Changes to a software system are inevitable. The catalyst for change can come from the market, new requirements, changes to business processes, technology advances, and bug fixes, among other things. Some view software architecture as inhibiting agility and would prefer to just let it emerge without up-front design. However, a good software architecture helps with both implementing and managing changes.

**Reorientation of SDLC w.r.t Architecture**

**Conclusions**