Introduction to software architecture

At the beginning of a development cycle for a complex application that combines multiple components,
we need to have a proper structure that represents all interactions happening within.

Software architecture is the term that defines such a structure. It helps understand the relationships between each major system component and simultaneously provides the necessary documentation for developers and clients.

Every project that combines multiple different components has to have a clear software architecture to define technical and structural requirements for the system.

It allows you to reduce possible risks when creating a project. If you need to complete your project and establish communications within a team in a short amount of time, you need to master software architecture.

What is software architecture?

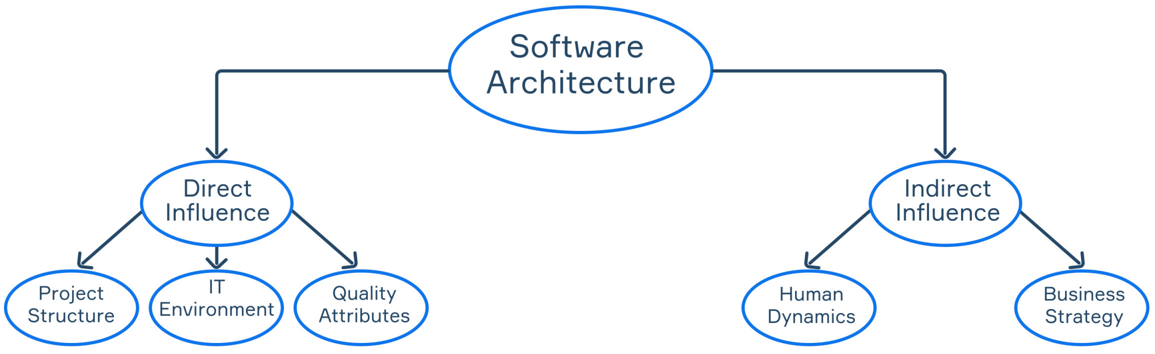
Basically, **software architecture or SA** is a pattern, which describes all inner components of the system and interactions between them.

It has spheres of influence that you can classify as Direct and Indirect.

By direct influence, you can make any changes to the project itself. Like improvements in security and quality attributes, optimising project structure, and any influence on used technologies.

Indirect influence means changes to the environment of a project. In the picture down below you can see how the classification goes.

SA directly influences **Project Structure, IT environment,** and **Quality Attributes.** While indirect influence encompasses **Human Dynamics** and **Business Strategy.**



Why would you need software architecture?

When working on a complex project, developers need to understand what they are doing, what they will do next, and what they are supposed to get in the end.

An architectural pattern that describes every connection within a system can help developers understand their project and adjust the workflow.

Here's a list of advantages of software architecture:

- It describes the system in a simple way without implementation details.
- It displays all working scenarios.
- It distinguishes functional and quality requirements.
- It improves the working and business environment.

If you properly distribute functions between the elements of SA, you could potentially save time for the developers working on implementing new features.

Also, it would be easier for the new team members to adjust to the workflow.

Then again, although creating a software architecture can make a real difference, it’s not always treated properly. Thus we experience a shortage of methods to represent SA and a lack of prioritization in making one.

Types of software architecture

You won't always need to create an absolutely unique software architecture, because there are lots of different pattern types created as optimal solutions to commonly occurring problems.

There are a few types of patterns like *Layered*, *Primary-Secondary*, *Peer-to-Peer*, etc.

To represent any pattern you could use schemes, models, diagrams, and even documents that collect every architectural decision (like **Architecture Decision Record (ADR)**).

Also, you can visualise SA through Unified Modelling Language or UML**. UML** is a visual language that combines different types of diagrams specified for individual purposes.

For a better understanding, here's a list of a few basic types of software architecture:

- **Application architecture –** description of patterns required to build an application. This type of architecture provides a team with a roadmap to build a well-structured app.
- **Information architecture –** description of the content organization, structure, and labeling in a way that allows users to conveniently access the information they need.
- **Database architecture** – a concept that focuses on the design, development, implementation, and maintenance of a database management system (DBMS). Simply put, it's an architecture that describes all the steps needed to realize and support a database and gives proper access to its functions for users and admins.
- **Network architecture** – description of a network system's physical and logical design. Basically, it depicts relations between network components and their logical functions.