In software engineering, project architecture is the fundamental organizational structure of a software system, acting as a blueprint that defines its components, their relationships, and how they interact to meet both business and technical requirements. It provides an abstract, high-level view of the system, focusing on its core structures, significant design decisions, and qualities like scalability and security while hiding implementation details. Architecture serves to manage complexity, coordinate components, and guide the development process by establishing a shared understanding and a set of constraints for the project.

Key aspects of software architecture:

* **Blueprint for a System:**

It's the plan or blueprint for an entire software system, outlining its high-level structure and how its various parts will function together.

* **Abstraction of Complexity:**

Architecture manages the complexity of a system by providing an abstract view that simplifies understanding and communication among stakeholders.

* **Defining Components and Relationships:**

It identifies the main software components, the rules for their interaction, and the communication mechanisms between them.

* **Addressing Requirements:**

Architecture ensures the system meets both functional requirements (what the system does) and non-functional requirements (such as performance, security, and reliability).

* **Guiding Development:**

It provides direction and structure for the development team, ensuring consistency and making it easier to maintain, extend, and optimize the system over time.

* **Balancing Trade-offs:**

Architects analyze and make decisions about competing quality attributes, such as performance versus security, and document the rationale behind these choices.

The Role of the Architect:

A software architect leads technical decisions and direction for a software design, ensuring it aligns with the project's business goals and user needs. They interact with stakeholders, analyze solutions, consider alternatives, and finalize a plan, documenting it in architecture documents with diagrams. This formalized plan is then handed off to engineering teams for implementation.

In software engineering, a [Data Flow Diagram (DFD)](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifNASO4bYBLp7DXt1v94r21J3Yj8xQ%3A1758073449180&q=Data+Flow+Diagram+%28DFD%29&sa=X&ved=2ahUKEwiF87H71d6PAxVO4TgGHdN1AFIQxccNegQIAxAB&mstk=AUtExfAZVsCjtf5EghpJ5p71dsx8U0UrJCIGFF-uHIsneCMWrDnnurkKB-WgUqjwilYstP8k8nuXZYrIEnonxMDS3fnxsanMNPbY068t2vKSV5zUZKigw2uj1j8byKZRyQLISm6_G1q1IgQFd7cku5DWO_6Huu-t2YoVpOf_Z3NEUo4h_vevAgDMTqwDRlnc8VrG7QH7gfD1hZqGvXHLrhfBxQ1ii2dNjat9Z7O0U1RAWFzk16Ds0uCtvkeKk1wamubkQpBEZsViz0I8kXafgQFsEpSi&csui=3) is a visual representation showing how data moves through a system or process. It uses symbols to illustrate data sources, transformations (processes), storage locations (data stores), and destinations (external entities). DFDs help software engineers understand complex systems, plan architecture, improve performance by identifying bottlenecks, and communicate data flow to stakeholders.

Key Components of a DFD

DFDs are composed of four main symbols:

* [**External Entity**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifNASO4bYBLp7DXt1v94r21J3Yj8xQ%3A1758073449180&q=External+Entity&sa=X&ved=2ahUKEwiF87H71d6PAxVO4TgGHdN1AFIQxccNegQIEhAB&mstk=AUtExfAZVsCjtf5EghpJ5p71dsx8U0UrJCIGFF-uHIsneCMWrDnnurkKB-WgUqjwilYstP8k8nuXZYrIEnonxMDS3fnxsanMNPbY068t2vKSV5zUZKigw2uj1j8byKZRyQLISm6_G1q1IgQFd7cku5DWO_6Huu-t2YoVpOf_Z3NEUo4h_vevAgDMTqwDRlnc8VrG7QH7gfD1hZqGvXHLrhfBxQ1ii2dNjat9Z7O0U1RAWFzk16Ds0uCtvkeKk1wamubkQpBEZsViz0I8kXafgQFsEpSi&csui=3)**:** A source or destination of data, such as a person, department, or another system.
* [**Process**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifNASO4bYBLp7DXt1v94r21J3Yj8xQ%3A1758073449180&q=Process&sa=X&ved=2ahUKEwiF87H71d6PAxVO4TgGHdN1AFIQxccNegQIFRAB&mstk=AUtExfAZVsCjtf5EghpJ5p71dsx8U0UrJCIGFF-uHIsneCMWrDnnurkKB-WgUqjwilYstP8k8nuXZYrIEnonxMDS3fnxsanMNPbY068t2vKSV5zUZKigw2uj1j8byKZRyQLISm6_G1q1IgQFd7cku5DWO_6Huu-t2YoVpOf_Z3NEUo4h_vevAgDMTqwDRlnc8VrG7QH7gfD1hZqGvXHLrhfBxQ1ii2dNjat9Z7O0U1RAWFzk16Ds0uCtvkeKk1wamubkQpBEZsViz0I8kXafgQFsEpSi&csui=3)**:** A step or function that receives input data and transforms it into output data.
* **Data Flow:** An arrow indicating the direction of data movement between components.
* [**Data Store**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifNASO4bYBLp7DXt1v94r21J3Yj8xQ%3A1758073449180&q=Data+Store&sa=X&ved=2ahUKEwiF87H71d6PAxVO4TgGHdN1AFIQxccNegQIFBAB&mstk=AUtExfAZVsCjtf5EghpJ5p71dsx8U0UrJCIGFF-uHIsneCMWrDnnurkKB-WgUqjwilYstP8k8nuXZYrIEnonxMDS3fnxsanMNPbY068t2vKSV5zUZKigw2uj1j8byKZRyQLISm6_G1q1IgQFd7cku5DWO_6Huu-t2YoVpOf_Z3NEUo4h_vevAgDMTqwDRlnc8VrG7QH7gfD1hZqGvXHLrhfBxQ1ii2dNjat9Z7O0U1RAWFzk16Ds0uCtvkeKk1wamubkQpBEZsViz0I8kXafgQFsEpSi&csui=3)**:** A location where data is stored or held, such as a database or file.

Purpose of DFDs

* **System Understanding:**

DFDs simplify complex systems by providing a clear, high-level overview of data movement.

* **Process Improvement:**

They help identify bottlenecks, redundancies, and inefficiencies in workflows, enabling better system design.

* **Communication:**

DFDs serve as a common visual language, improving communication and collaboration among team members and with project users.

* **Analysis:**

They are used in system analysis, business management, and software development to plan new systems or optimize existing ones.

Levels of DFDs

DFDs are often developed in layers or levels:

* [**Context Diagram**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifNASO4bYBLp7DXt1v94r21J3Yj8xQ%3A1758073449180&q=Context+Diagram&sa=X&ved=2ahUKEwiF87H71d6PAxVO4TgGHdN1AFIQxccNegQIShAB&mstk=AUtExfAZVsCjtf5EghpJ5p71dsx8U0UrJCIGFF-uHIsneCMWrDnnurkKB-WgUqjwilYstP8k8nuXZYrIEnonxMDS3fnxsanMNPbY068t2vKSV5zUZKigw2uj1j8byKZRyQLISm6_G1q1IgQFd7cku5DWO_6Huu-t2YoVpOf_Z3NEUo4h_vevAgDMTqwDRlnc8VrG7QH7gfD1hZqGvXHLrhfBxQ1ii2dNjat9Z7O0U1RAWFzk16Ds0uCtvkeKk1wamubkQpBEZsViz0I8kXafgQFsEpSi&csui=3)**(Level 0):**

A high-level overview showing the entire system as a single process, its external entities, and the main data flows.

* [**Level 1 Diagram**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifNASO4bYBLp7DXt1v94r21J3Yj8xQ%3A1758073449180&q=Level+1+Diagram&sa=X&ved=2ahUKEwiF87H71d6PAxVO4TgGHdN1AFIQxccNegQISxAB&mstk=AUtExfAZVsCjtf5EghpJ5p71dsx8U0UrJCIGFF-uHIsneCMWrDnnurkKB-WgUqjwilYstP8k8nuXZYrIEnonxMDS3fnxsanMNPbY068t2vKSV5zUZKigw2uj1j8byKZRyQLISm6_G1q1IgQFd7cku5DWO_6Huu-t2YoVpOf_Z3NEUo4h_vevAgDMTqwDRlnc8VrG7QH7gfD1hZqGvXHLrhfBxQ1ii2dNjat9Z7O0U1RAWFzk16Ds0uCtvkeKk1wamubkQpBEZsViz0I8kXafgQFsEpSi&csui=3)**:**

A more detailed breakdown of the system, showing the main processes within the system and the data flows between them.

* **Subsequent Levels:**

Further breakdown of processes into more detailed sub-processes, creating a hierarchy of diagrams to progressively elaborate on the system's data flow.

In software engineering, a visual diagram is a graphical representation of a software system's components, structure, and behavior, using symbols and connections to explain complex concepts like architecture, data flow, and user interactions. These diagrams serve as blueprints and communication tools, simplifying code and making it easier to understand, design, and troubleshoot systems for both technical and non-technical stakeholders.

Why visual diagrams are important

* **Communication:**

They provide a common visual language for development teams, product managers, and other stakeholders, fostering clearer communication and collaboration.

* **Understanding:**

Diagrams abstract complex codebases into understandable visual representations, allowing for quick comprehension of how different parts of a system work together.

* **Design and Planning:**

They help in the design process by providing a roadmap for building scalable, modular, and secure systems.

* **Troubleshooting:**

By offering a clear view of the system's design, diagrams can highlight potential bottlenecks and issues, aiding in troubleshooting efforts.

* **Onboarding:**

New hires can quickly grasp the architecture of a new system by reviewing its diagrams instead of sifting through thousands of lines of code.

Common types of visual diagrams

* [**UML Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=UML+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQILBAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:**

A standardized language for visualizing software systems, with types including:

* + [**Class Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=Class+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQILhAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:** Show the static structure of a system, including classes, their attributes, and operations.
  + [**Sequence Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=Sequence+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQIMBAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:** Illustrate how objects interact with each other over time in a specific sequence.
  + [**Use Case Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=Use+Case+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQIMhAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:** Depict the relationships between users (actors) and the functions (use cases) of the system.
  + [**Deployment Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=Deployment+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQIMRAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:** Show the physical hardware and software components of a system and how they are deployed.
* [**Architecture Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=Architecture+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQILRAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:**

Provide high-level blueprints of a system's components, how they are structured, and how they interact.

* [**Swimlane Diagrams**](https://www.google.com/search?sca_esv=3dd5e06a00554f5a&cs=0&sxsrf=AE3TifPl2WP3mT-nWf9GuoTQYrwTus-l9w%3A1758073531813&q=Swimlane+Diagrams&sa=X&ved=2ahUKEwi_kLqi1t6PAxXUxDgGHWaOCOwQxccNegQILxAB&mstk=AUtExfCO4l9o8gger1qrjQRU5l_7CzO5Mo1IrYeqrHfI__K9cCcEsUQKm6A4R5WftamGi0-rx_WEH4wREOHZPgziXoTf8-1fY4jyn2itgb1D_kTIUcP8IJ21lR6HcdLgnbhX4mn0ceoFl86_v3YFNRiDzryIRXZI-fJ1iO7HwNolZ3VTfB6-a3h9OztG4Qyspx0QMBxvpEqjlgtK5x0I602Xa1dZio9DaGncyUavEKtIPi70Z6dmpXb67eoE8FomY1J2ZSzKKOgZ9PJoqLOR2O-hYB7L&csui=3)**:**

Used in project planning to assign responsibilities to different teams or departments for each stage of the development process.

In software engineering, UI wireframes are basic, blueprint-like visual representations of a software's user interface (UI) that focus on the structure, layout, and placement of content and features, rather than visual design details like colors or fonts. They serve as a low-fidelity sketch or "skeleton" for a webpage or application, helping designers, developers, and stakeholders align on functionality and content structure before investing time in high-fidelity designs and coding.

Key Characteristics of Wireframes

* **Low-Fidelity:**

Wireframes are generally simple and lack visual polish, using basic shapes and grayscale.

* **Structure and Layout Focused:**

They emphasize how different elements, such as buttons, navigation, and content blocks, are arranged on the screen.

* **Content Organization:**

Wireframes demonstrate the logical flow of information, guiding users through the interface intuitively.

* **Functional Blueprint:**

They act as a plan, outlining the placement and purpose of features before the actual building or coding begins.

Why Wireframes Are Important

* **Early Alignment:**

They provide a common understanding of the product's core functionality for the entire team and stakeholders.

* **Focus on Functionality:**

By removing visual distractions, wireframes allow teams to concentrate on the fundamental aspects of the user experience.

* **Iterative Design:**

They are ideal for the initial stages of development, allowing for quick iteration and feedback on the product's structure.

* **Cost-Effective:**

Making changes at the wireframe stage is far less time-consuming and expensive than making them during later stages of development, according to this YouTube video.