Software Architecture

🖼️ View

A **view** is a **representation** of a coherent **set of architectural elements**, as written and read by system stakeholders.

A **view** consists of a **representation** of **set of elements** and the **relations** among them.

🧩🖼️ Module View

A **module view** is the **representation** of the structure, documented according to a template in a chosen notation, and used by some system stakeholders.

Module views are excellent for showing someone the structure of a project.

- Who does what, which teams are assigned to which parts of the system, and so forth.

- In any module structure, the elements are modules of some kind (perhaps classes, layers, or merely divisions of functionality, all of which are units of implementation).

- Modules are assigned functional responsibilities; there is less emphasis in these structures on how the software manifests at runtime.

💀 Skeletal System

A **skeletal system** is the **infrastructure** (how the elements initialize, communicate, share data, access resources, report errors, log activity) built before the system's functionality has been created

Once an **architecture** has been defined, it can be analyzed and prototyped as a **skeletal system**

🧩🖼️ Module Structures

Modules are assigned to **specific computational responsibilities**, and are the basis of work assignments for programming teams.

Modules are **design time entity**.

🧱 Architectural Structures

A **structure** is a set of elements held together by a relation.

There are three important categories of architectural structures:

1. Module
2. Component and Connector
3. Allocation

A **structure** is the set of elements, as they exist in software or hardware.

🔗 Component and Connector (C&C) Structures

Structures focuses on **the way elements interact with each other** at runtime to carryout the system's functions.

Components are **runtime entity**.

🔎 Allocation Structures

Describe the **mapping** from software structures to the **system's environment**.

Organizational, developmental, installation, and execution.

Software Architecture

🚩 Importance of Software Architecture

1. An architecture will inhibit or enable a system's driving quality attributes
2. The analysis of an architecture enables early prediction of system's qualities
3. A documented architecture enhances communication among stakeholders
4. An architecture defines a set of constraints on subsequent implementation
5. An architecture is the key artifact that allows the architect and project manager to reason about cost and schedule
6. An architecture can be created as a transferable, reusable model that form the heart of a product line
7. Architecture-based development focuses attention on the assembly of components, rather than simply on their creation
8. An architecture can be the foundation for training a new team member

📝 Definitions

→ Is the **set of structures** needed to reason about the system, which comprise software **elements**, **relations**, and **properties** of both.

→ It refers to the **high-level structures** of a software system and the **discipline** of creating these structures. It involves a set of significant **decisions** about the organization of the software system, including the **selection of structural elements** and their **interfaces**, by which the system is composed and behavior as specified in collaborations among those elements.

→ Is the **high-level design** and organization of software in which we make important **decisions** regarding its **overall structure**, such as the **relationships between components**, **data flow** patterns, and the **mechanism for communication** between different parts of the system.

→ **Architecture** is the **fundamental organization** of a software system embodied in its **components**, their **relationships** to each other or to the **environment**, and the **principles** guiding its design and evolution.

Architecture Includes Behavior

This behavior embodies how elements interact with each other, which is clearly part of the definition of architecture

What are these early design decisions embodied by software architecture?

- Will the system run on one processor or be distributed across multiple processors?

- Will the software be layered? If so, how many layers are there? What will each one do?

- Will components communicate synchronously or asynchronously? Will they interact by transferring control, data, or both?

🎈 The work-breakdown structure in turn dictates:

- Units of planning, scheduling, and budget
- Inter-team communication channels
- Configuration control and file-system organization
- Integration, test plans, and procedures

Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap

Architectural Design

📊 Software Product Line

A **software product line** or family is a **set of software** systems that are all **built using** the same set of **reusable assets**

Chief among these assets is the **architecture** that was designed to handle the needs of the entire family or product line

🧩 Using Independently Developed Components

- Commercial off-the-shelf components
- Open source software
- Publicly available apps
- Networked services

Are examples of **interchangeable software components**

🚗 Architectural Drivers

• Design Purpose
• Quality Attributes
• Primary Functionality
• Architectural Concerns
• Constraints

Architectural Drivers also guide cost and schedule estimation, team formation, risk analysis, and implementation

🎯 Goals

Technical: achieving low and predictable latency in a video game or an e-commerce website

Nontechnical: keeping the workforce employed, entering a new market, meeting a deadline

START

📑 Architectural Requirements

🏛️ Architectural Design (AD)

Design is a **translation**, **from** the world of **needs** (requirements) **to** the world of **solutions**, in terms of structures composed of code, frameworks, and components

📖 Architectural Documentation

Some level of preliminary documentation (or sketches) of the structures should be created **as part of architectural design**

However, this activity refers to the creation of a more formal document from these sketches

✔️ Architectural Evaluation

As with documentation, if your project is nontrivial, then you owe it to yourself and to your stakeholders to evaluate it, to **ensure** that the **decisions** made are appropriate to address the critical requirements

🚧 Architectural implementation/conformance checking

Your major responsibility during implementation is to ensure **conformance** of the **code** to the **design**

If developers are not faithfully implementing the architecture, they may be undermining the qualities that you have designed

🪣 Architectural Design Include

To create a reliable, secure, and efficient product, the attention is needed to **architectural design** which includes:

• Overall organization
• Server organization
• How the software is decomposed into components
• The technologies that is used to build the software

🔼 More info about Architectural Design

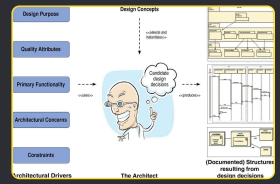
🔄 Types of Architectural Design Changes

1. **Local Change**: modifying a **single element**.

2. **Non-local Change**: **multiple element** modifications but leaves the underlying architectural approach intact.

3. **Architectural Change**

A good architecture is one in which the **most common changes are local**, and hence easy to make.



✍️ Architectural Design Info

In **architectural design**, we make **decisions** to **transform** design purpose, requirements, constraints, and architectural concerns —**architectural drivers**— into **structures**

Architectural design is a key step to **achieve** your product and project **goals**

A **good design** is one that **satisfies** the **drivers**

🔠 The main outputs of the AD phase are the

• Architectural Design Document (ADD)
• Software Project Management Plan for the DD phase (SPMP/DD)
• Software Configuration Management Plan for the DD phase (SCMP/DD)
• Software Verification and Validation Plan for the DD Phase (SVVP/DD)
• Software Quality Assurance Plan for the DD phase (SQAP/DD)
• Integration Test Plan (SVVP/IT)

Logical Model

The **logical model** produced in the SR phase, **structures the problem and makes it manageable**. The **physical model** does the same for the solution

Physical Model

The **physical model** is used to **produce** a **structured set of component specifications** that are consistent and complete.
Each specification defines the functions, inputs, and outputs of the component.

Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap

Architect & Designs

- **Performance**: You must manage the time-based behavior of elements, their use of shared resources, and the frequency and volume of inter-element communication

- **Modifiability**: Assign responsibilities to elements so that the majority of changes to the system will affect a small number of those elements

- **Reusability**: Restrict inter-element coupling, so that when you extract an element, it does not come out with too many attachments to its current environment

Architectural Patterns

Architectural patterns guide the architect and focus the architect on the **quality attributes** of interest, in large part by restricting the vocabulary of design

Interaction Design Methods

- Defining Use Cases

A use case is a written description of how users will perform tasks on an application or product

Interaction Design

Is the design of **interactions** between **users** and **products**

The 5 dimensions of interaction design

The 5 dimensions of interaction design is a useful model to understand what interaction design involves

1D: Words

- Like button labels should be meaningful and simple to understand

- Words should communicate information to users, but not too much information to overwhelm the user

2D: Visual representations

Like images, typography and icons that users interact with

3D: Physical objects or space

- A laptop, with a mouse or touchpad?

- Smartphone, with the user's fingers?

4D: Time

- It is the amount of time a user spends interacting with the product

- Can users track their progress, or resume their interaction some time later?

5D: Behavior

- This includes the mechanism of a product: How do users perform actions on the website? How do users operate the product?

- It also includes the reactions—for instance emotional responses or feedback—of users and the product.

The Role of the Architect

- **Defining and designing the software architecture for a project**, by selecting technologies, frameworks, and patterns to create a robust and scalable system

- **Collaborating with stakeholders**, such as project managers, business analysts, and software developers, to understand their needs and requirements, ensuring that the architecture aligns with the project's goals

- **Providing technical leadership and guidance to development team**, mentoring new developers, and sharing knowledge for best practices and architectural principles

- **Identifying and addressing potential technical risks and challenges**, proactively assessing the system's architecture to identify any potential issues

- **Creating and maintaining comprehensive software architecture documentation**, including design decisions, architectural patterns, and technical standards, to serve as a reference for the stakeholders throughout the product roadmap

Architect

Architect is worried about strategies to achieve all goals

User is concerned that the system is fast, reliable, and available when needed

Role types

Element Internals Design

Conducted as part of the element development activities

Detailed Design

The term "**detailed design**" is often used to refer to the design of the internals of modules

Solution Architects

Focuses primarily on identifying **technological solutions** to potential **challenges** a business encounters

They ensure the overall technical solution they create for a specific business problem fulfills the enterprises' requirements as well as end-user needs

Enterprise Architects

Maintain and **update** a company's IT **networks** regularly and in the long term to ensure optimal performance

They analyze business needs to ensure the overall design and implementation align with the business goals

Data Architects

Responsible for defining procedures, policies, tools, and models the software development team uses to create, organize, store, and retrieve organizational data

Also create standards for the collection, storage, and migration of information from existing data structures or legacy systems

Cloud Architects

Responsible for creating, designing, managing, and executing the organization's cloud computing infrastructure

They develop and deploy applications to the cloud and keep track of cloud activities such as taking account of maintenance schedules and monitoring resource usage

Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap

Types of Systems & Architectures

Architectural Drivers (AD)

- Design purpose
- Quality attributes
- Primary functionality
- Architectural concerns
- Constraints

Quality Attributes

Measurable or **testable properties** of a system. It shapes the architecture significantly.

The best way to discuss, document, and prioritize quality attributes is **scenarios** (scenarios describes the system's response to some stimulus).

Examples:
- Performance
- Usability
- Reusability
- Portability
- Learnability
- Security

Architectural concerns

Additional aspects that need to be considered as part of architectural design but that are **not** expressed as **traditional requirements**.

There are several different types of concerns:

1. General concerns.
2. Specific concerns.

Constraints

These constraints may take the form of:

- Mandated technologies.
- Other systems which your system needs to integrate.
- Laws and standards that must be complied with.
- Abilities and availability of your developers.
- Deadlines that are non-negotiable.
- Backward compatibility with older versions.

Questions:

Q: Write any 3 architectural drivers?

Q: What do we mean by domain?

Q: Write any 6 quality attributes?

Q: What is or explain the architecture in the diagram?

Q: Match the steps of ADD 3.0 diagram?

Q: Compare between greenfield and brownfield systems?

At early stages, architectural design will determine the following:

- Key approaches for achieving architectural drivers.
- The gross work-breakdown structure.
- The choices of tools and skills.
- Technologies needed to realize the system.

Architectural Design

Design Concepts

- Design is not random, but rather is planned, intentional, rational, and directed.
- To help achieve modifiability, modularity, high cohesion, and low coupling.
- To help achieve high availability and avoid having any failure.
- To help achieve security and limit the access to critical resources.

Architectural Styles

Such as "Pipe and Filter" and "Client Server"

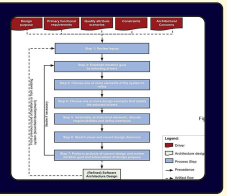
Defined types of **components** and **connectors**, that are **useful for structuring** an application either logically or physically

Attribute-Driven Design 3.0 (ADD)

Same as SDLS we have DevOps or Agile, in architecture design we have methodologies to follow in designing an architecture, like this ADD 3.0.

For your info: From Architectural Drivers we get input, then in the first step, we review these inputs.

Q: Match the steps?



More Info

Prototype
- To explore the domain.
- To explore new technology.
- To place something executable in front of customers.
- To elicit rapid feedback.
- To explore more quality attributes.

Reference Architecture. A **Blueprint** that **provides** an overall **logical structure** for particular type of applications.

A **reference architecture** is a **reference model**, mapped onto one or more architectural patterns.

4 + 1 View Model of Software Architecture
- Logical View
- Process View
- Development View
- Physical View
- Use Cases or Scenarios (+1)

Examples of application types

- **Data Processing Apps:** Data driven apps, that process data in batches, without user intervention during the processing.
- **Transaction Processing Apps:** Data-centered apps, that process user requests and update information in a database.
- **Event Processing Systems:** Apps where system actions depend on interpreting events from the system's environment.
- **Language Processing Systems:** Apps where the users' intentions are specified in a formal language then processed and interpreted by the system.

Existing System (Brownfield)

Architecture design for an existing system may occur for different purposes, most obvious is **maintenance**

System Type

Greenfield Systems

Greenfield Systems for Mature Domains

When **designing an architecture** for a system built **from "scratch"**.

This type of system is **well known and understood** because **there is an** established **infrastructure** of tools, **technologies**, and associated **knowledge** base.

Examples of Mature Domains:

- Traditional desktop apps.
- Interactive mobile apps.
- Enterprise applications.

Greenfield Systems for Novel Domains

In the case of novel domains, it is more **challenging** to **establish** a precise **roadmap**, because reference **architectures** may be **few** or don't **exist**.

Layered Architecture

Definition

Organizes the system into a **set of layers** (or abstract machines), each provides a set of services.

When a layer interface changes, only the adjacent layer is affected.

Advantages

- Allows **replacement of entire layers** as long as the interface is maintained.
- **Redundant facilities** (e.g., authentication) can be provided in each layer to increase the dependability of the system.

Disadvantages

- **Separation between layers is often difficult**, a high layer may have to interact with lower layers, rather than through the layers in between.
- Performance can be a problem because of **multiple levels interpretation** for a service request.

Repository Architecture

Definition

Sub-systems **exchanges data**.

This may be done in two ways:
- Shared data is a **central database** or repository, and **accessed** by sub-systems.
- Each sub-system maintains **its own database**, and **passes data** to other sub-systems.

When a **large amount of data is shared**, the **repository model** is most **commonly used**, for its **efficient** data sharing **mechanism**.

Advantages

- **Components can be independent** — they do not need to know the existence of other components.
- **Changes** made by one component can be **propagated** (distributed) to other components.
- All **data** can be **managed consistently**.

Disadvantages

- The **repository is a single point of failure**, so problems in the repository affect the whole system.
- May be **inefficient** in **organizing communication** through the repository.
- **Distributing** the **repository** across several computers may be **difficult**.

Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap

Design Concepts, Structures & Interfaces

Module Structures

Logical and **static elements**, that **exist at development time**, such as **files**, **modules**, and **classes**.

Component and connector (C&C) structures

Dynamic elements that **exist at runtime**, such as **processes** and **threads**.

Allocation structures

Both software elements (Modules or C&C) **and non-software elements**, that **may exist at development time and runtime**, such as file systems, hardware, and development teams.

🔎 1.1 Identification of Design Concepts

Once you have more clarity regarding the design concepts, you still need to identify alternatives — that is **design candidates**

Leverage your own knowledge and experience:
If the system you are designing is similar to other systems you have designed, better to begin with the design concepts that you have used before.

Step 1:
Identify and Select Design Concepts

🏢 2.1 Structures

The architecture of a software system is composed of a set of **structures**, which can be grouped into **three major categories**:

- Module structures
- Component and connector (C&C)
- Allocation structures

🎯 1.2 Selection of Design Concepts

Create a table to list the **pros** and **cons** of each alternative and select one of the alternatives, based on the criteria and drivers.

Step 2:
Produce Structures

✨ 2.2 Instantiating Elements

The **instantiation of deployment patterns** involves the **identification** and **specification** of **physical elements**.

For example, if you are using a **Load-Balanced Cluster pattern**, instantiation may involve:

- Identifying the number of replicas
- The load-balancing algorithm
- The physical location of the replicas.

Externally Developed Components Instantiation

The instantiation of these components may or may not imply the creation of new elements.

Cost Benefit Analysis Method (CBAM)

CBAM is a method that **guides the selection of design alternatives** using a **quantitative approach**.

Step 3:
Defining Interfaces

External interfaces

Interfaces **from other systems** t required by the system that you are developing, and interfaces that are provided by your system to other systems.

🔗 3.1 Interfaces

The **externally properties** of elements, that establish a contractual specification, **allowing elements to collaborate** and exchange information.

Two categories of interfaces: **external** and **internal**.

Internal Interfaces

Interfaces between the elements.

To identify relationships and interface details; we need to understand how elements exchange information at runtime.

Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap

Software Architecture Design Methods & ADL

📑 Tasks in Refining Architecture

- **Identifying** design **mechanisms** and **elements**.
- **Performing** operation **analysis**.
- **Incorporating** existing **design elements**.
- **Structuring** the implementation **model**.
- **Describing** the **runtime** architecture and **distribution**.
- **Reviewing** the architecture.

👁️ Viewpoint

A viewpoint is a **set of patterns**, templates, and conventions to **create** a **specific** type of **architectural view**.

It **defines stakeholders** and **addresses** their **concerns** through guidelines, and template.

Viewpoints include functional, information, concurrency, development, deployment, and operational viewpoints.

👤 Architectural perspective

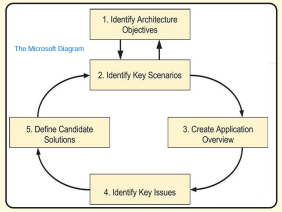
Activities, tactics, and guidelines **to ensure** a system **meets quality properties** across its **architectural views**.

Key perspectives in Rozanski and Woods's book include security, performance, scalability, availability, resilience, and evolution.

📝 Questions:

Q: Write any 3 general models for software architecture?

Q: Compare between logical and physical architecture?



🔄 Major Activities in The Process of Software Architecting

1. Define Requirements: Establishes system needs.

2. Create Logical Architecture: Focuses on an architecture that is largely technology-independent.

3. Create Physical Architecture: Focuses on the architecture with technology considerations.

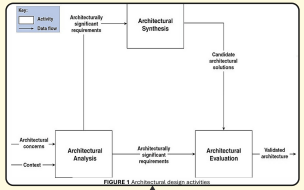


FIGURE 1 Architectural design activities

Software Architecture Design Methods

🧠 Rational Unified Process (RUP)

Phases

Iterative development is a key feature of RUP, where iterations occur across four sequential phases:

1. Inception: In this phase, the **project is conceived** and **feasibility is evaluated**.

2. Elaboration: In this phase, many **necessary aspects** to successfully **develop** the project are **handled**. One of these aspects is the architecture design.

3. Construction: In this phase, the system is **built iteratively**.

4. Transition: In this phase, the completed system is **transitioned**

Guidance in RUP

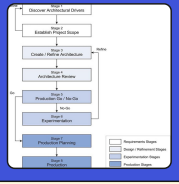
Strong focus on architectural concerns such as defining the system context and establishing a system structure.

Detailed guidance on both logical and physical architecture.

✳️ Architecture-Centric Design Method (ACDM)

A software architecture development method, that **covers the complete life cycle** of the architecture.

This iterative method consists of **8 stages**:



🅰️ The Five General Models for Software Architecture

- ADD 2.0
- Siemens 4 views
- RUP's 4+1 Views
- Business Architecture Process and Organization (BAPO)
- Architecture Separation of Concerns (ASC)

📏 ADL Definition

A formal language used to **describe** the **architecture** of a software system.

Provides **notations** for **specifying** the **components**, their **interactions**, and the overall structure of the system.

Supports both **graphical** and **textual representations, enabling** developers and architects to **communicate**.

Supports architectural reasoning and decision-making.

Used for **documenting** software **architecture**.

Enables automated tools in development.

📑 ADLs focus on documenting:

- Computational components
- Interactions among components
- Properties of the architecture

Architecture Description Languages (ADL)

📍 Popular ADLs

- **ACME**: Used for system architecture interchange and analysis.
- **Aesop**: Focuses on customizable software architecture styles.
- **Rapide**: Enables system simulation and execution traces.
- **Wright**: Specializes in describing system's behavior, especially connectors.
- **Darwin**: Focuses on the specification of distributed systems.

✨ ADL Features

- Graphical notations for visual representation
- Formally defined textual notations for precise specifications

Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap

Analysis in the Design Process

🔫 Tactics-Based Analysis

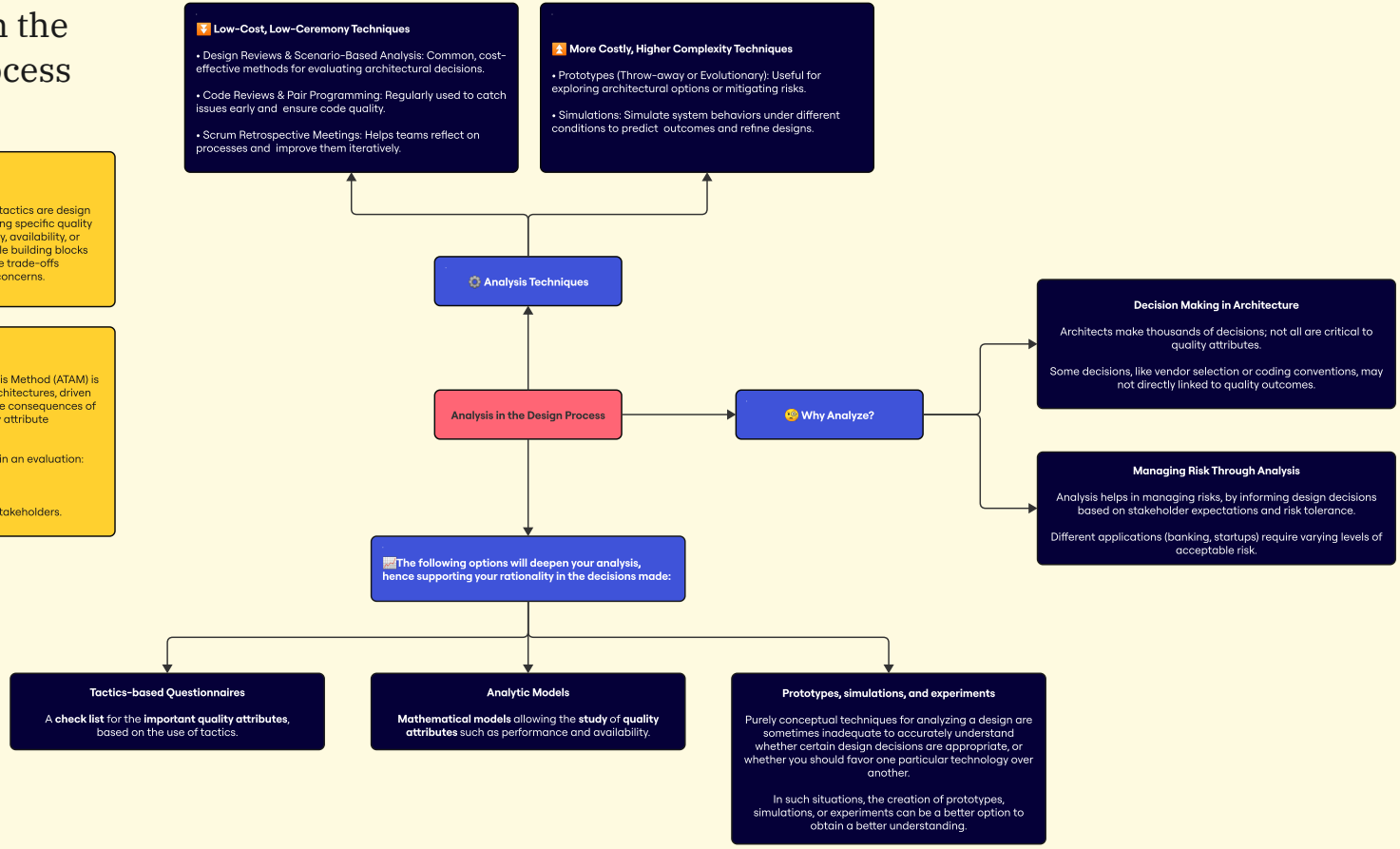
In the context of software architecture, tactics are design decisions or strategies aimed at achieving specific quality attributes such as performance, security, availability, or scalability. Tactics are low-level, reusable building blocks that architects can apply to manage the trade-offs associated with different architectural concerns.

🪢 Scenario-Based Design Reviews

The ATAM—Architecture Tradeoff Analysis Method (ATAM) is an established method for analyzing architectures, driven by scenarios. Its purpose is to assess the consequences of architectural decisions in light of quality attribute requirements and business goals.

The ATAM brings together three groups in an evaluation:

- A trained evaluation team.
- An architecture's "decision makers" .
- Representatives of the architecture's stakeholders.



Designed by: Soft Conclusion
YouTube link: Soft Conclusion - YouTube
Miro link: Software Design & Architecture MindMap