CS2703 SOFTWARE ARCHITECTURE

Unit -1

Architectural Views and Quality Attributes

* Software Architecture

→ The software architecture of a system is the set of structures needed to reason about the system, relations among them, & the properties of both.

→ Architecture speaks about components, their attributes and their inter relationship within a system

→ serves as the backbone for implementing a complex system

* Architecture as an Abstraction

→ software architecture is analogous to entity relation model & schema in a database.

(i) System Architecture → hardware, software components

(ii) Enterprise Architecture → Organizational functions, Business units, workflows

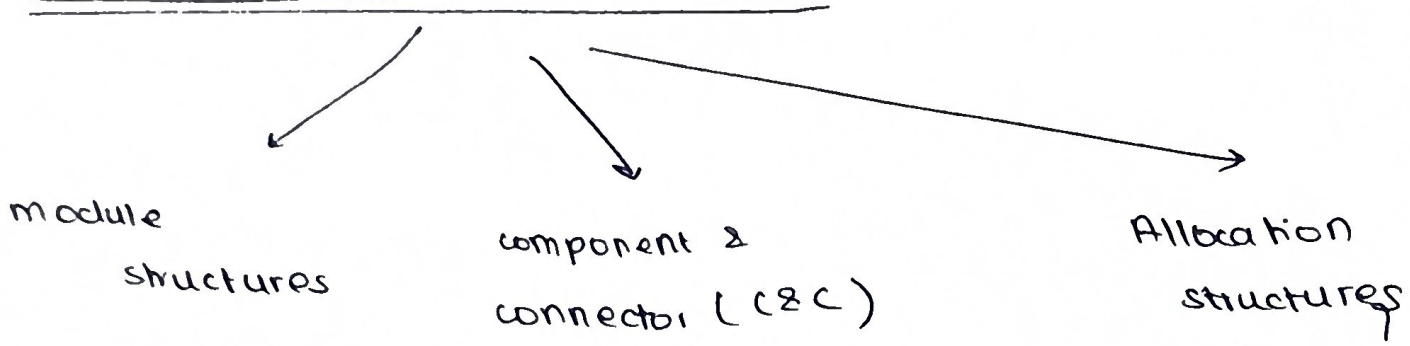
(iii) Structures - has actual components

- module structures, component & connector, allocation structres

(iv) Views - representation of the components

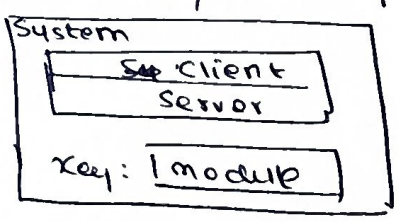
- client-server, decomposition view

* Software Architecture Structures



A. Module Structures → modules and their relationships

(i) Decomposition - divide the system into smaller modules, helping with modifiability by enabling encapsulation & configuration control



(ii) Uses - shows dependencies where a module relies on another to function

(iii) Layers - Organizes modules hierarchically, supporting incremental development

(iv) Class : Describes classes and objects within object-oriented design

(v) Data Model - manages data relationships.

B. **Component & Connector (C&C) Structures** → focus on runtime elements

(i) Service - defines services that may run concurrently, crucial for performance & interoperability
→ uses service coordination mechanisms - like SOAP simple object access protocol

(ii) Concurrency - indicates parallel processes or threads.
- determine opportunities for parallelism, threads & the locations where resource contention may occur

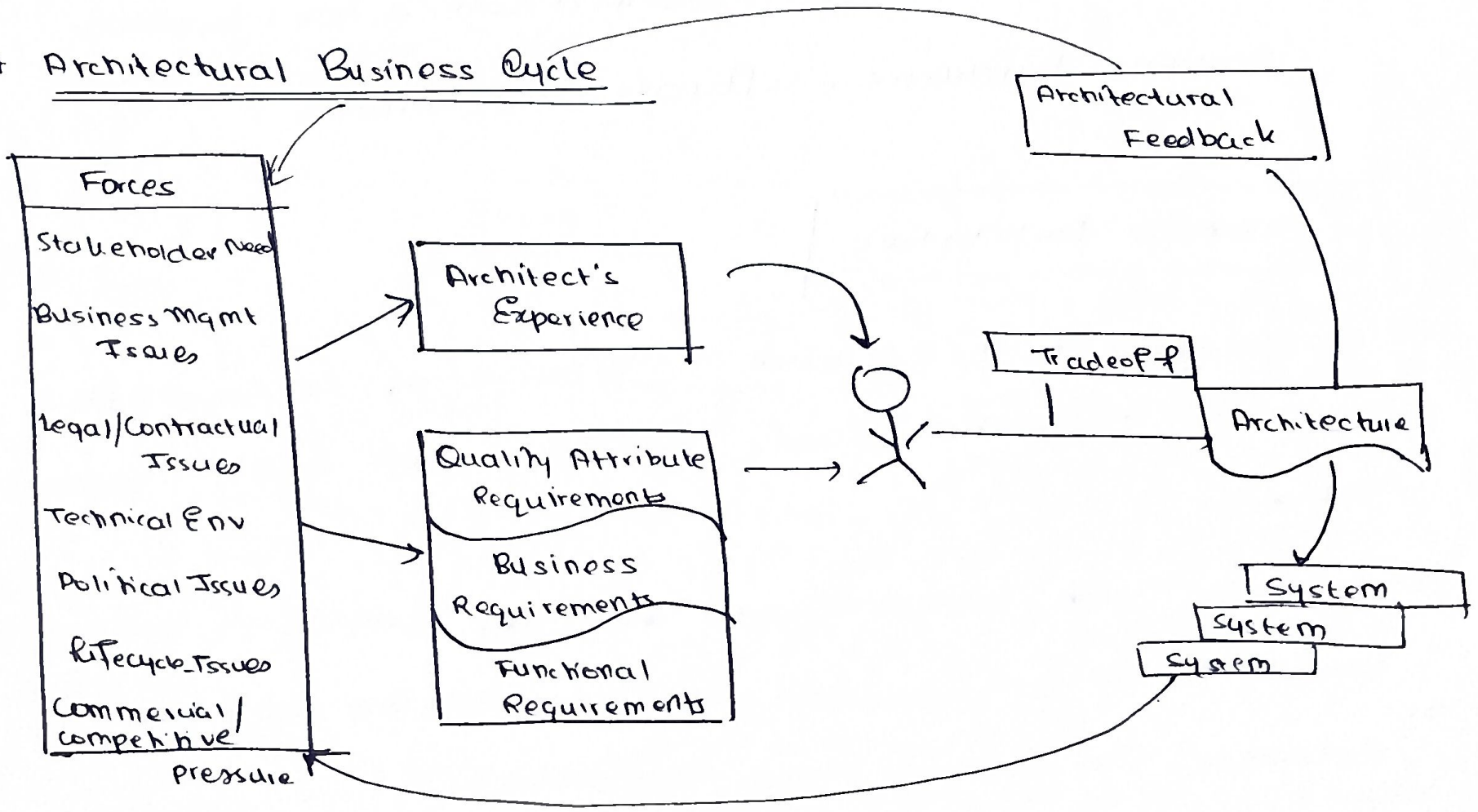
C. **Allocation Structures** → how software elements are mapped to hardware or organizational units

(i) Deployment - assigns software components to hardware

(ii) Implementation - Details where code modules are stored
→ mapping to file structures in the system's development, integration, configuration control environments

(iii) Work Assignment - organizes work based on expertise & resource management

* Architectural Business Cycle



* System and Enterprise Architecture

1. **System Architecture**

→ a representation of a system in which there is a mapping of functionality onto hardware & software components.

→ a mapping of software architecture onto the hardware architecture

→ a concern for the human interaction with these components

→ allows for reasoning about additional qualities such as power consumption, weight & physical footprint

→ shows how functionality is implemented & how the various processors interact through the exchange of messages on the network.

∴ Software architecture is concerned with a total system, including hardware, software and humans

2. **Enterprise Architecture**

→ a description of the structure & behavior of an organization's processes, information flow, personnel, and organizational subunits

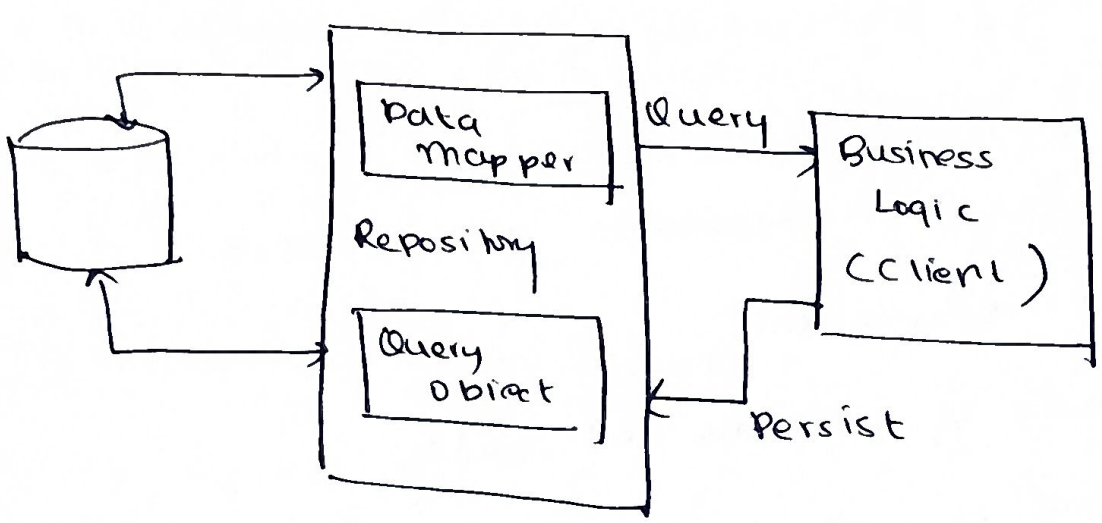
→ concerned with software, and how the software is used by humans to perform business processes and the standards that determine the computational environment.

* Architecture Patterns

① Layered – when the 'uses' relationship among software elements is strictly unidirectional, a systems of layers emerges

→ A layer is a coherent set of related functionality

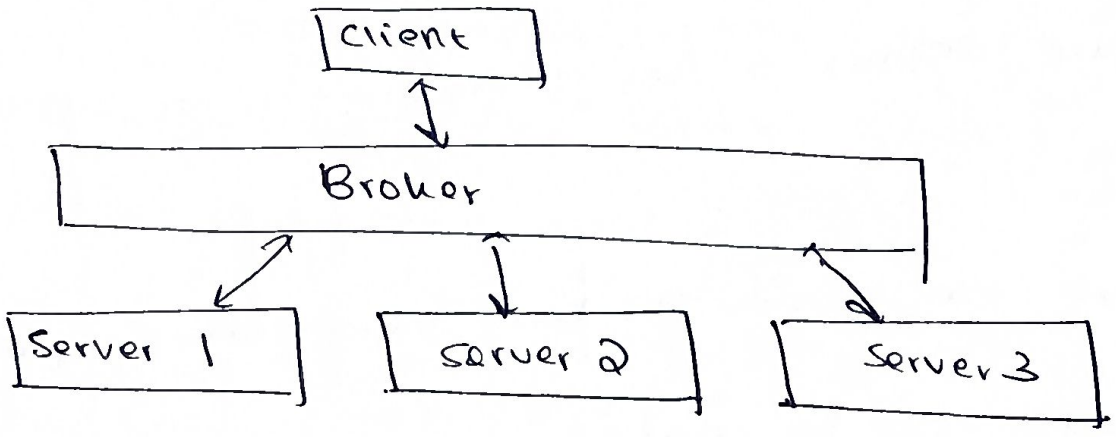
② Shared - Data – has components and connectors that create, store and access persistent data.



has

Presentation
Business
Persistence
Database
} layers

③ Client-Server Patterns – components are the clients and servers and the connectors are protocols and messages that share among each other to carry out the system's work



* Functional & Non-Functional Requirements

through the design of whatsapp

Functional
- send a message
- receive a message
- read message
- groups
- images

Non-Functional - scale
- availability
- latency

* Key Aspects of a Reference Architecture

→ The reference architectural design must be good. Must have the following features:

- portability / interoperability
- reliability
- well-defined - components and relations
- version independent
- right choice of patterns and styles
- right choice of structures & views

PRUVRR

* Importance of Architecture Quality

→ An architecture will inhibit or enable a system's driving quality attributes.

→ An architecture defines a set of constraints on subsequent implementation

→ An architecture can provide the basis for evolutionary prototyping.

→ An architecture can be created as a transferable, reusable model

→ An Architecture can be created as a transferable, reusable model

→ Architecture-based development focuses attention on the

assembly of components, rather than simply their creation

→ The architecture dictates the structure of an organization or vice-versa.

→ It provides inputs for cost budgeting and scheduling in software project execution

* Common Architectural Styles

(i) Pipe and Filters
(ii) Object orientation
(iii) Event-based Trigger Systems
(iv) Layered Systems
(v) Repositories
(vi) Table-driven

Other Styles → distributed systems
→ domain specific systems
→ process control systems
→ state transition
→ main program - sub routine

* Architectural Structure (Mostly a repetition of pages 223)

① Key considerations for structure
→ Gross organization and global control structure
→ Protocols for communication

* Role of high-level programming languages in SA

→ shift from machine language to assembly language to high level programming language over the decades

→ High level languages allow more sophisticated programs to be developed, and patterns of data to be used.

→ Introduction of modules to provide protection for related procedures and data structures - encapsulation in OOPs

→ allows for separation of a module's specification from its implementation

* Enabling System Quality Attributes

1. If your system requires high performance

2. If modifiability is important

3. If your system must be highly secure

4. If scalability is important

5. If your products need the ability to deliver incremental subsets

6 If your want elements from your system to be reusable

* Predicting System Quality Attributes

→ It is possible to make quality predictions about a system based solely on an evaluation of its architecture.

→ The design analysis must be a standard part of the architecture development process.

→ used to ensure that an architecture will deliver its prescribed benefits

→ must be abstract enough that most non-technical people can understand it

→ must overcome constraints & support decision making

Check the following

→ If it is influencing the organizational structure in a tve & effective way.

→ If it is enabling evolutionary prototyping

→ If it is improving cost and schedule estimato

→ If it is supplying a transferable, reusable model

→ If it is allowing incorporation of independently developed components

→ If it is restricting the vocabulary of design alternatives and serves as a rolo model for training other modules

* Software Architecture for project life cycle

A. waterfall

B. Iterative

C. Agile

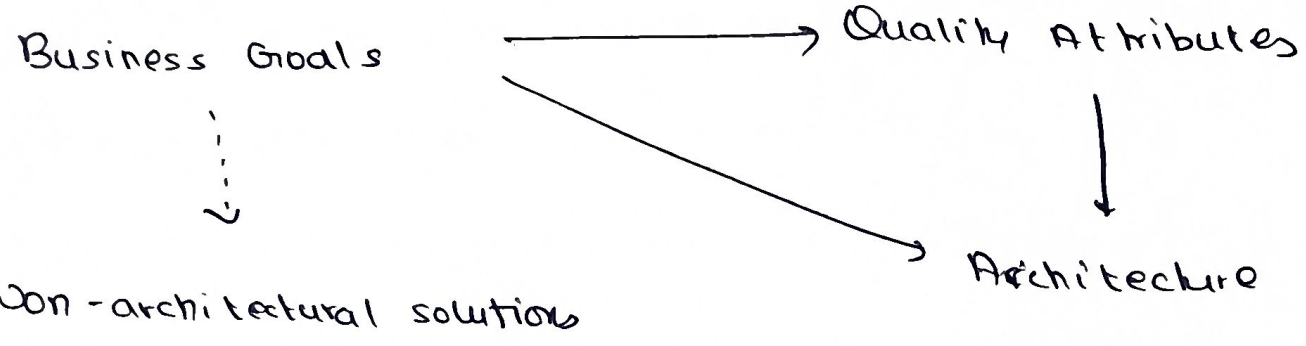
D. Model-based development

* Software Architecture for Business

→ Systems are created to satisfy the business goals of one or more organizations

→ Organizations want to make a profit, capture market, stay in business, make stockholders happy

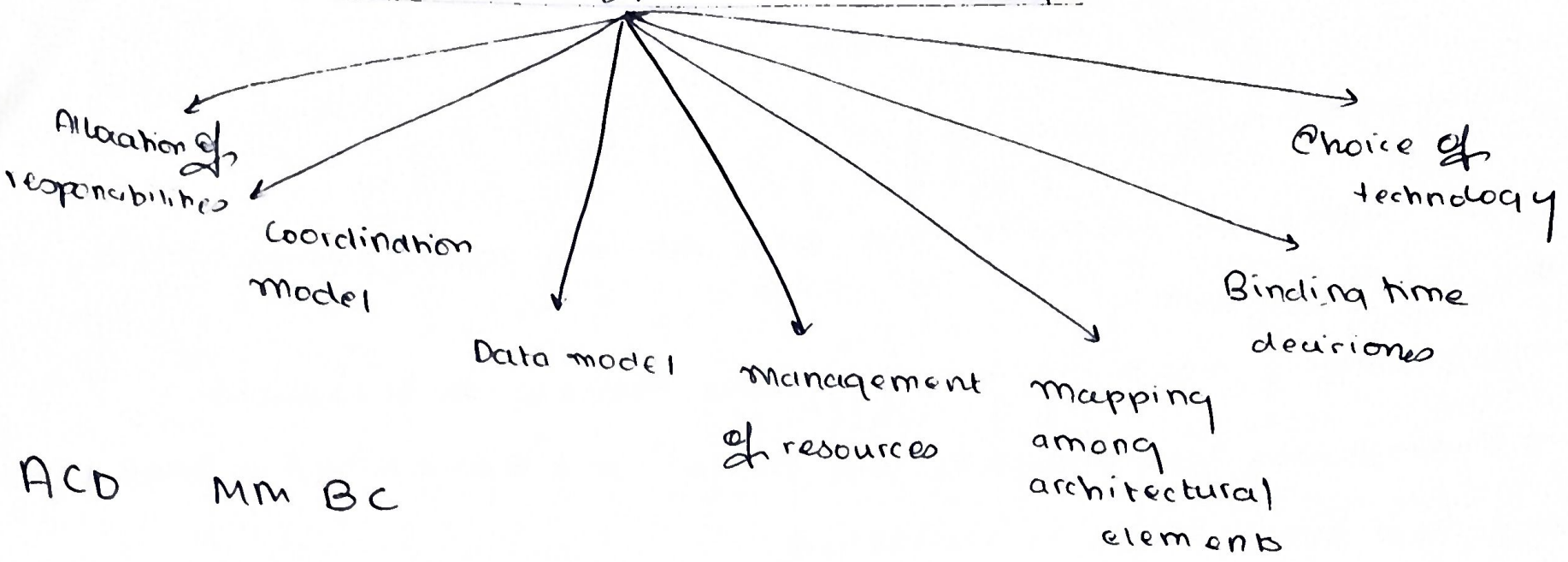
→ Architects need to understand who the vested organizations and what their goals are.



Examples of Business Goals : (i) maximize reusability and reduce cost

* Stakeholders

* Seven Categories of Design Decisions



ACD MM BC

(1) Allocation of Responsibilities

→ Identifying the important responsibilities, including basic system functions, architectural infrastructure, and satisfaction of quality attributes

→ Determining how these responsibilities are allocated to non-runtime and runtime elements - namely modules, components, and connectors

(2) Coordination model

→ Identifying the elements of the system that must coordinate, or those prohibited from coordinating

→ Determining the properties of the coordination such as timeliness, concurrency, completeness, correctness and consistency

→ Choosing the communication mechanisms

→ Important properties of the communication mechanisms include:

(i) stateful vs. stateless

(ii) synchronous vs. asynchronous

(iii) guaranteed vs. non-guaranteed delivery

(iv) performance-related properties such as throughput & latency.

(3) Data Model

→ choosing the major data abstractions, their operations and properties

→ includes determining how data items are created, initialized accessed, persisted, manipulated, translated and destroyed.

④ **Management of Resources**

→ Identifying the resources that must be managed, and determining the limits for each

→ Determining which system elements manage each resource

→ Determining how resources are shared and the arbitration strategies employed when there is contention

→ Determining the impact of saturation on different resources.

⑤ **Mapping among Architectural Elements**

→ assign / map runtime elements and modules to each other

(i) assign runtime elements to processors

(ii) assignment of items in the data model to data stores

(iii) mapping of modules and runtime elements to units of delivery

⑥ **Binding Time Decisions**

(i) allocation of responsibilities → build-time selection of modules

(ii) coordination model → runtime negotiation of protocols

(iii) resource management → system can accept new peripheral devices plugged in at runtime, after which the system recognizes them and downloads and installs the right drivers automatically

+1 - app choice

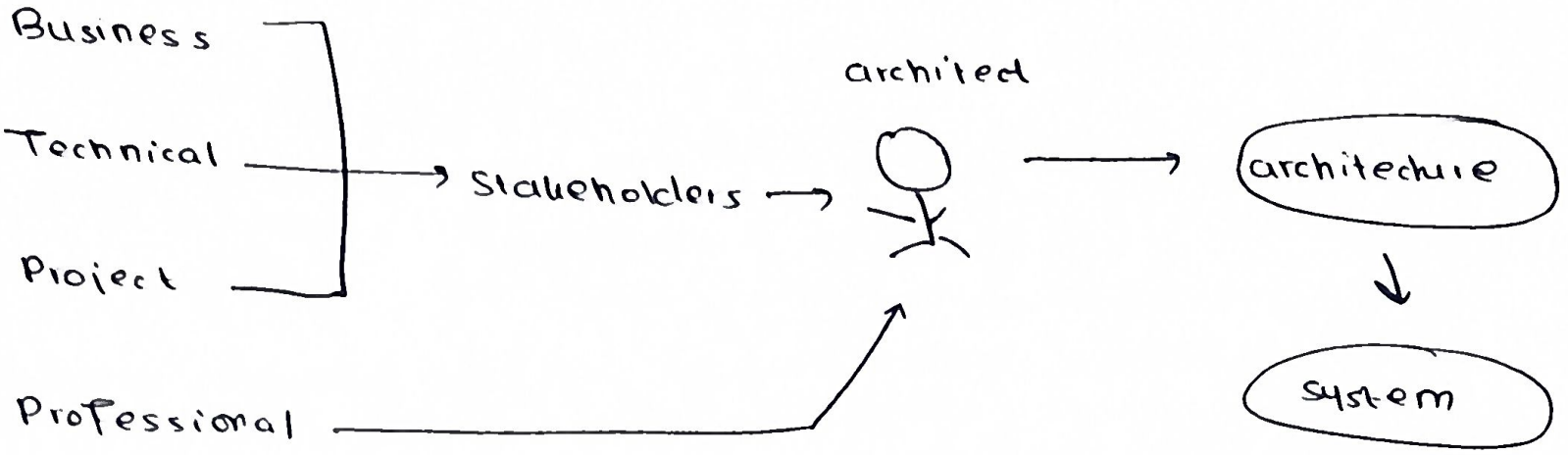
⑦ **Choice of Technology** → decide which technologies are available → are the IDEs, simulators, testing tools adequate to proceed?

→ Determining internal, external familiarity - contractors?

→ are there side effects?

→ is the new technology compatible.

* Influencers for Architecture



A. Technical Context - Architecture choices impact future systems by making it easier for customers to get reliable, cost-effective systems based on the same architecture

B. Project Context - Architecture affects how the organization is structured and outlines the parts of software that need to be created and assembled to build the system.

C. Business Context - The architecture can affect the business goals of the developing organization. A successful system built from an architecture can enable a company to establish a foothold in a particular market segment.

D. Professional Context - The process of system building will affect the architecture's experience with subsequent systems

* Requirements for a Software Architecture

Functional

Quality - scalability, reliability

Constraints - A design decision with zero-degrees of freedom

* Tactics → a specific design decision used to address a particular quality attribute like performance, reliability or security. It is smaller in scope than patterns and directly address the required challenge. Tactics are reusable and focus on implementing targeted solutions within the system.