* Software Architecture

Set of principle design decisions made about the system.

* Perspective Architecture
As-intended architecture.

* Descriptive Architecture
As documented architecture (eg UML)

* Architectural Drift

Introduction of principle design decision into systems descriptive architecture that are not included in perspective architecture do not voilate perspective design decisions.

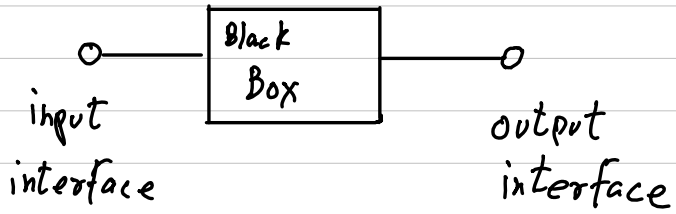
* Architectural erosion

Introduction of design decisions into systems descriptive architecture that voilate the design decisions of perspective architecture.

* Component in Architecture

Elements that do Processing
have data

Black boxes that do work




Self sufficient chunks of code that do work

* Connector in Architecture

A software connector is an architectural building block tasked with effecting and regulating interactions among components.

* Architectural styles

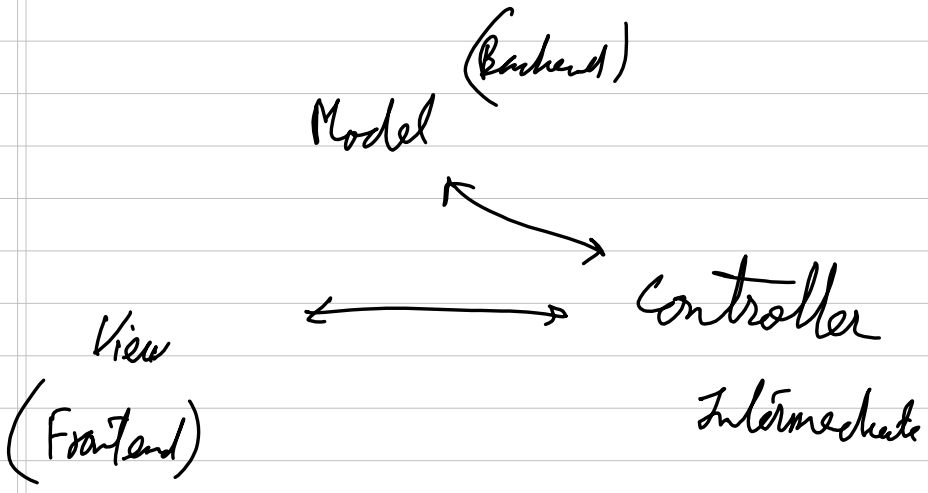
Named collection of architectural decisions that are applicable in given development context

* Architectural Pattern

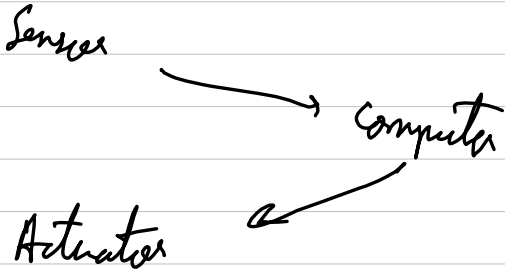
Set of architectural design decisions that are applicable to a recurring design problem

* 3 Tier Pattern

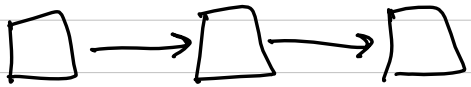

* MVC Pattern




* Sense compute control



* Batch Sequential Architecture



Process a batch of data sequentially
Wait till batch is full
eg share market

* Pipe and filter
Continuous stream of data


stream processing
low latency

Both have tight coupling

# Event Based

Independant components asynchronously emit and recieve events

Network busses.

Everything occurs as soon as possible

Nothing is waiting on anything else.

React to events

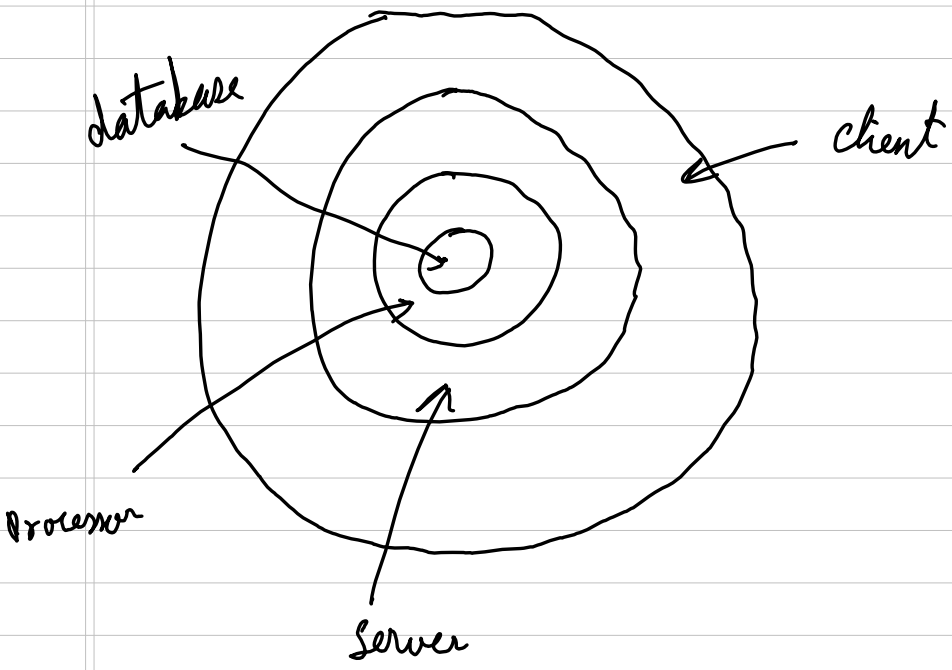
Highly scalable → loose coupling

Distributed applications

Useful for IoT systems

# Layered System

Each level exposes an interface API that can be used by next level




Changes in 1 layer affect only 2 layers

# Implicit Invocation

Raise flag / broadcast event

Dont know which will react to which event

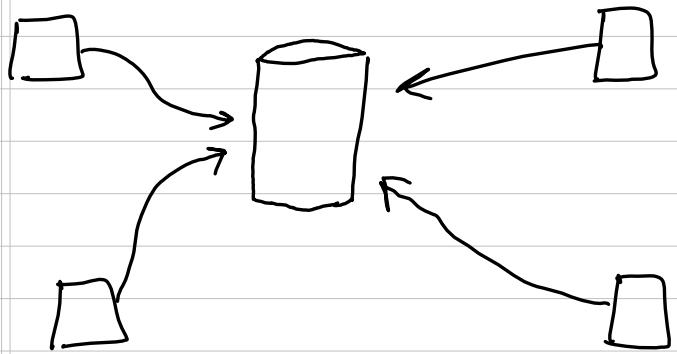
Components need not know others existance

Components can subscribe to events of interest

# Repository (Blackboard style)

Central data structure (Blackboard)

Every component store their results in blackboard.



No direct interaction

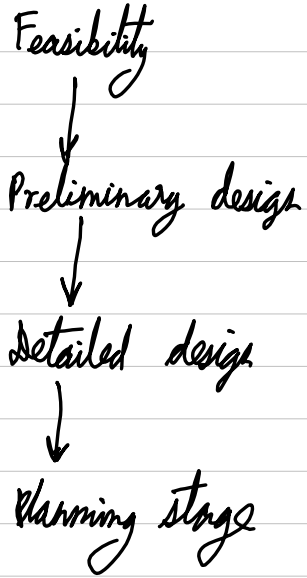
loose coupling

# Interpreter style

Parses and executes commands

Dynamic behaviour possible

# Design Process



# C2 style

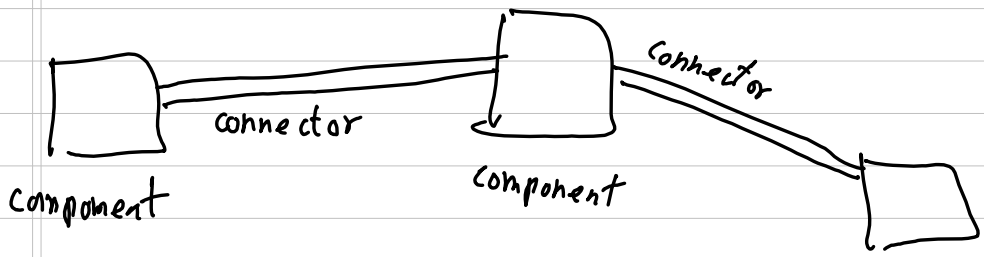
Component Connector style

Components are independant and dont interact with other components

Connectors facilitate the communication

loose coupling → high scalable

! Connectors can become complex




Decoupling

Flexibility

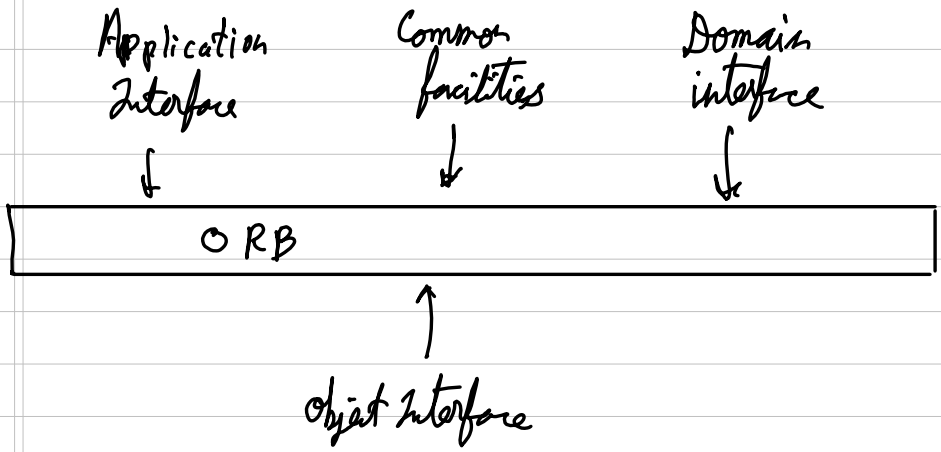
Interoperatibility between different languages

Maintainability due to independance of components

# CORBA

Common Object Request Broker Architecture

Hetrogenous hosts running hetrogenous languages.




Interoperability