

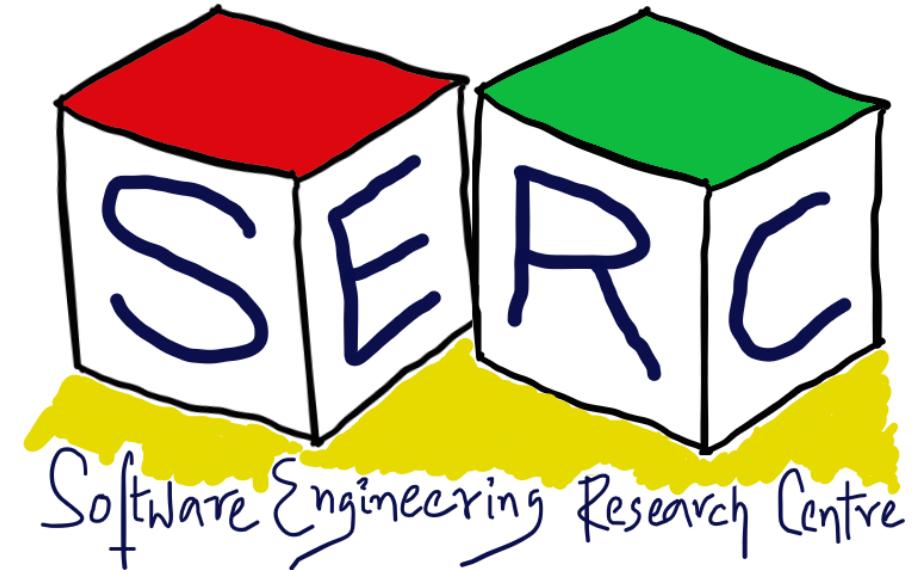
# Introduction to Software Architecture

**CS6.401 Software Engineering**

Dr. Karthik Vaidhyanathan

[karthik.vaidhyanathan@iiit.ac.in](mailto:karthik.vaidhyanathan@iiit.ac.in)

<https://karthikvaidhyanathan.com>



# Acknowledgements

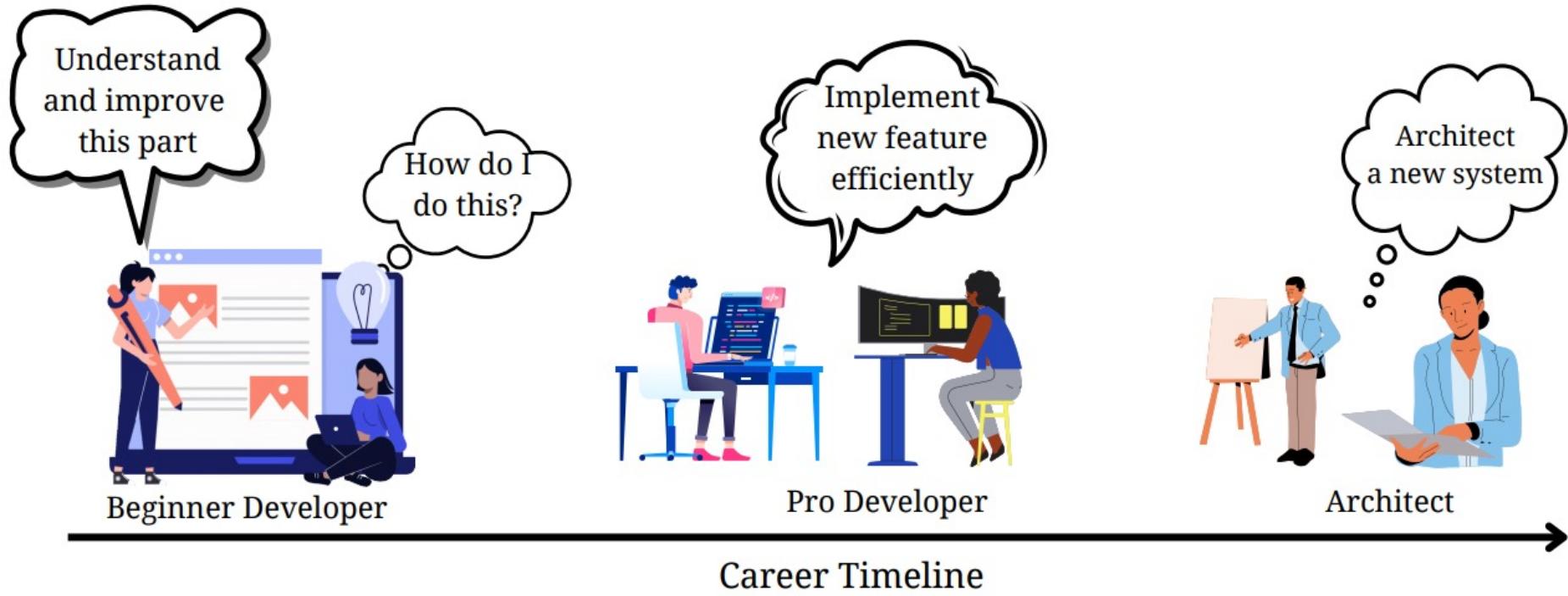
The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

## Sources:

1. Introduction to Software Architecture, Henry Muccini, University of L'Aquila
2. Software Architecture in Practice, Len Bass, 3<sup>rd</sup> edition
3. Software Architecture (SE Course), Alessio Gambi, Saaraland University, Germany
4. Software Architecture Design Reasoning Workshop, Antony Tang, ISAPS 2018



# The Journey



## This Course

Modeling

Refactoring

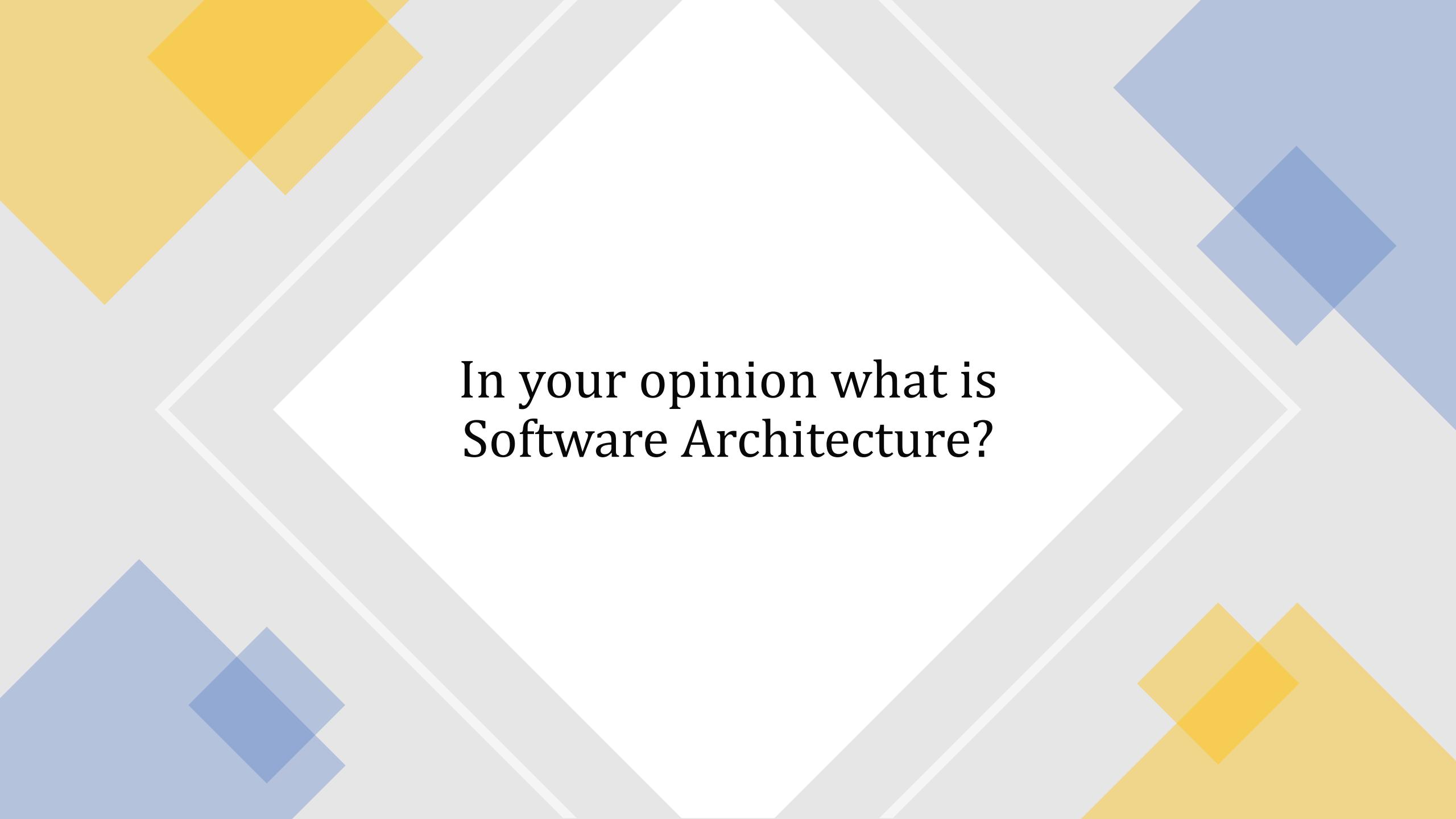
Design Principles

Design Patterns

Architectural Framework

Architectural Patterns





In your opinion what is  
Software Architecture?

# What will you do if you want to build a house?

- Family of 6
- Independent house
- Land of 40 cents
- Accommodate 6 + 2
- .....



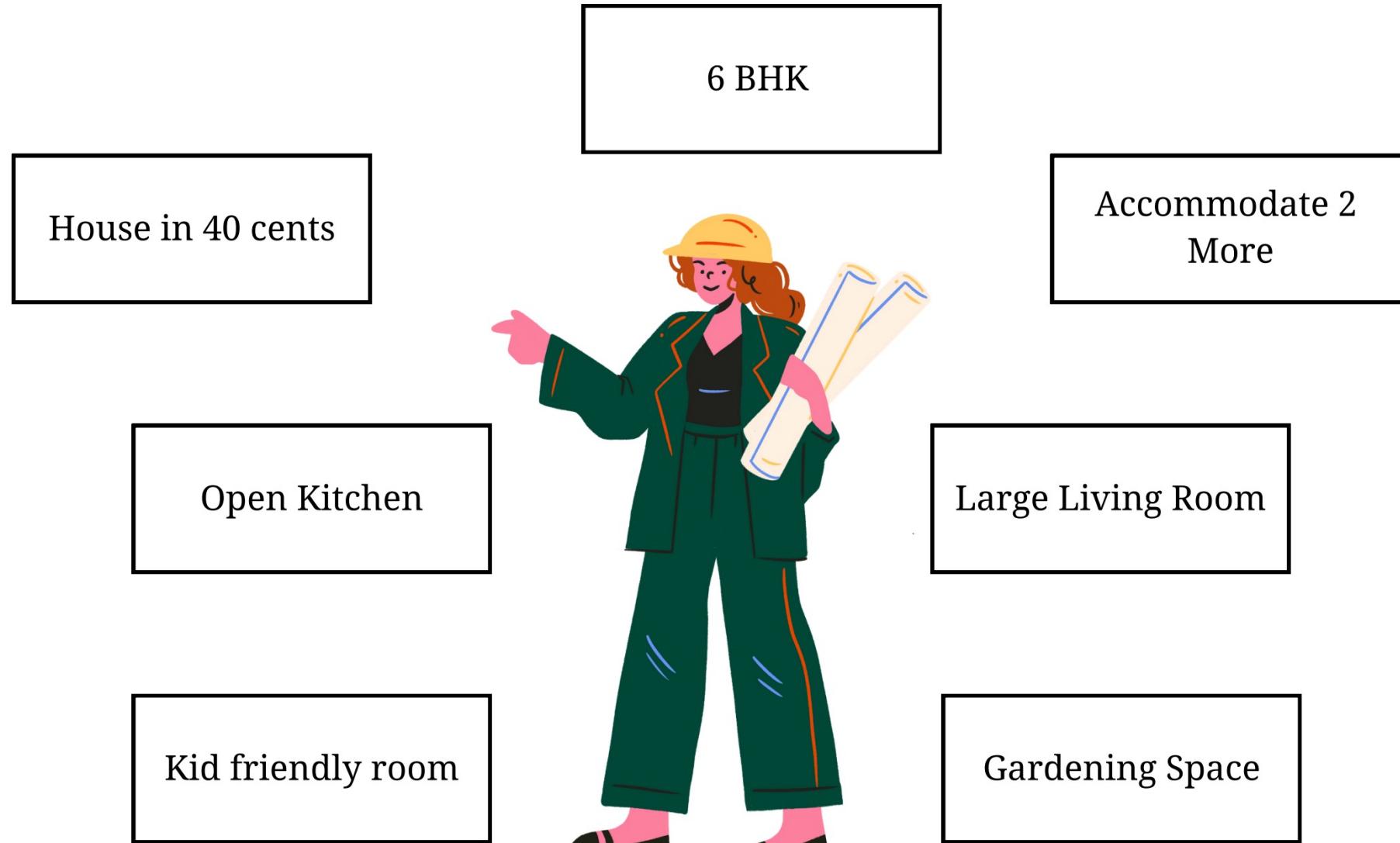
# Give the requirements!!



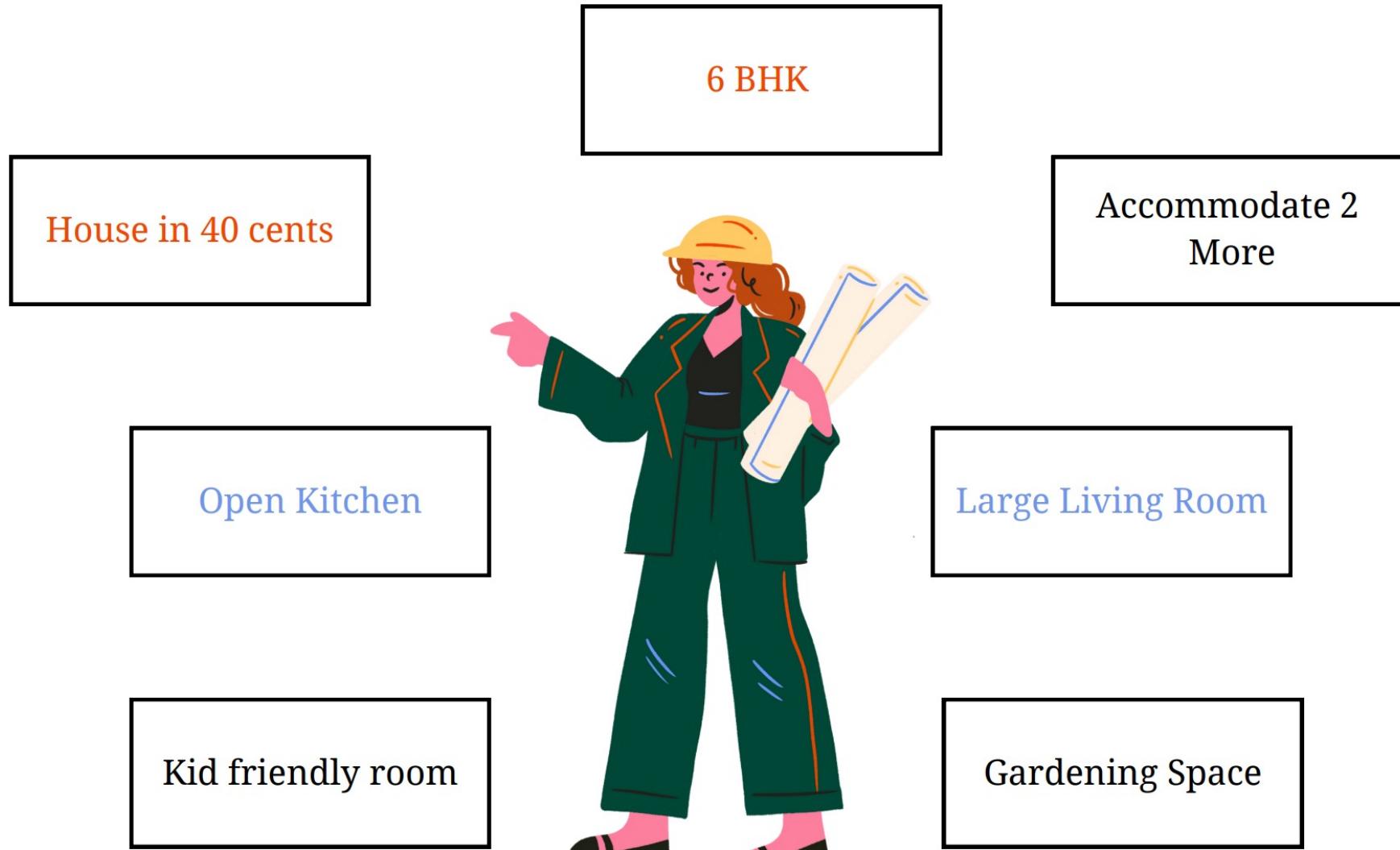
# Here you go!!! – There are always multiple Stakeholders



# Let me list all the requirements



# Let me see Constraints and Significant Requirements



# Time for Some Decisions

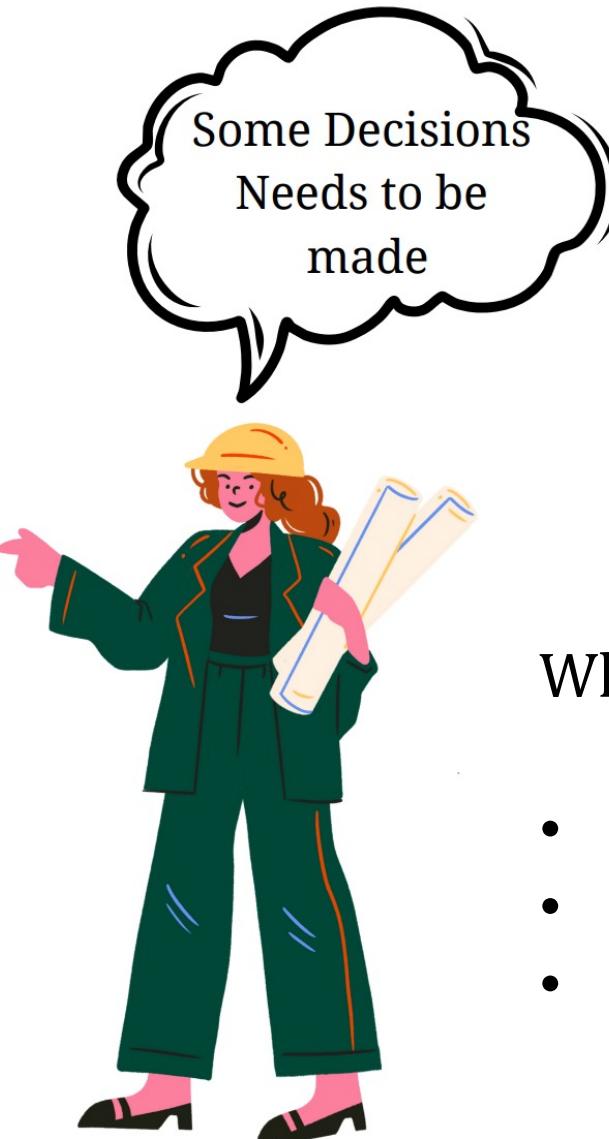
What material to be used?

- Budget
- Weather Condition
- ...

How to balance size?

- Living room size
- Kitchen Size
- ...

**Design Decisions!!**



What about window type?

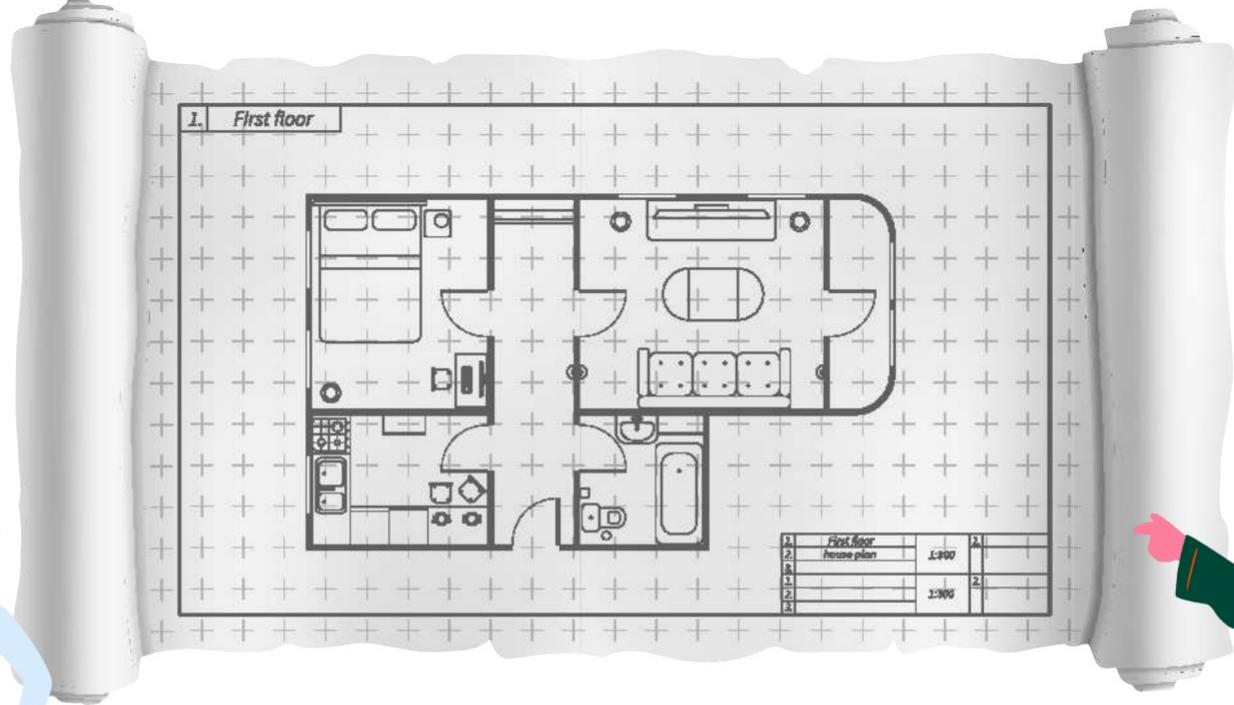
- French Window
- Normal window
- Consider window location
- ...

What about stair elevation?

- Roof length
- Type of people
- ...

# High level Plan Ready

This is good  
but can you provide  
3D view for  
others?



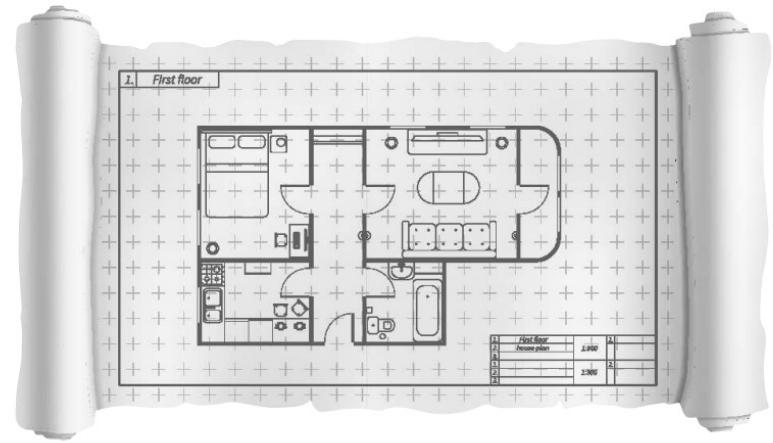
Floor Plan, each room, stairs,  
doors, size, how they connect,  
etc....



# Multiple Views for different set of people



3D View



2D View



Interior View

**Views and Viewpoints**

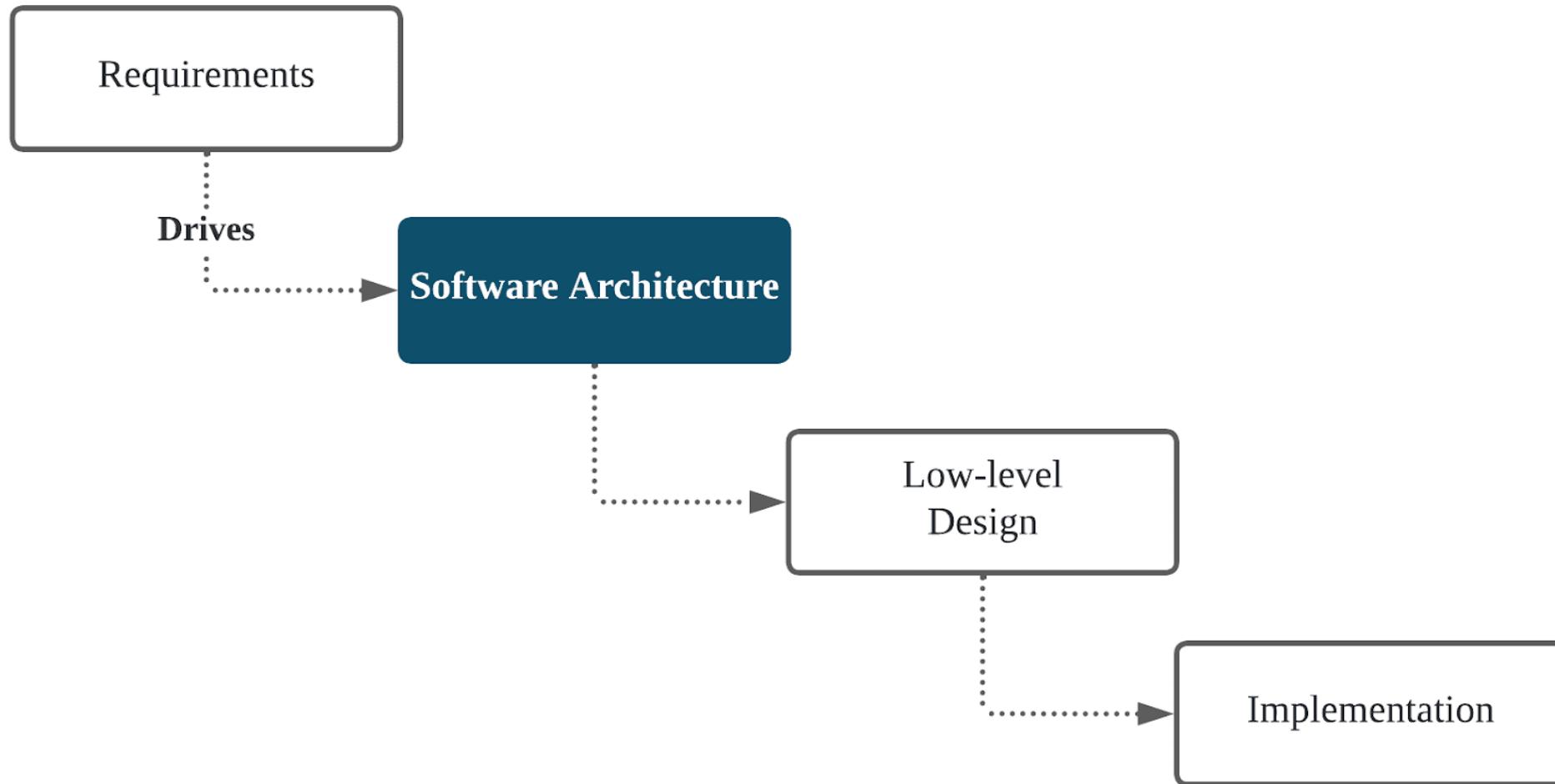
# Construction happens and the house is ready!





Let's take this to Software

# Where does Software Architecture come into Picture?



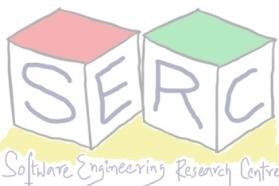
# Software Architecture: Is that it?

*“Software is **not limited by physics**, like buildings are. It is **limited by imagination**, by design, by organization. In short, it is limited by properties of people, not by properties of the world. We have **met the enemy, and he is us**”*



Martin Fowler, **Who needs an Architect?** IEEE Software, 2003

Ralph Jhonson



# What is Software Architecture?

# Software Architecture Definitions

Many different definitions exist for software architecture!

Check: <https://sei.cmu.edu/architecture/definitions.html>

# Software Architecture Definitions

- Garlan and Shaw, '93:

*Architecture* for a specific system may be captured as “**a collection of** computational **components** - or simply components - together with a description of the interactions between these components - the **connectors** ”

- Bass et al.:

“The software architecture of a program or computing system is the **structure or structures** of the system, which comprise **software elements**, the **externally visible properties** of those elements, and the **relationships** between them.”

# Software Architecture

ACM SIGSOFT

SOFTWARE ENGINEERING NOTES vol 17 no 4

Oct 1992 Page 40

## Foundations for the Study of Software Architecture

Dewayne E. Perry

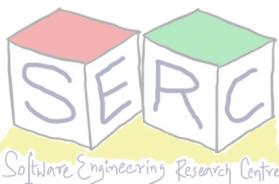
AT&T Bell Laboratories  
600 Mountain Avenue  
Murray Hill, New Jersey 07974  
[dep@research.att.com](mailto:dep@research.att.com)

Alexander L. Wolf

Department of Computer Science  
University of Colorado  
Boulder, Colorado 80309  
[alw@cs.colorado.edu](mailto:alw@cs.colorado.edu)

© 1989,1991,1992 Dewayne E. Perry and Alexander L. Wolf

**Software Architecture = {Elements, Form, Rationale}**



# Elements, Form and Rationale

## Elements (Components and Connectors)

1. Three types: data, processing and connecting elements
2. They form the foundational pieces of a software architecture

## Form (Architectural Patterns/Styles)

1. Weighted properties and relationships
2. Properties indicate the properties of the elements and weight indicates relevance/preference
3. Relationships constrain the placements of elements (how they interact, how they are organized, etc.)

## Rationale (Design Decisions)

1. Captures the motivation for choice of architectural style, elements and form

# Software Architecture

The Software Architecture is **the earliest model of the whole software system** created along the software lifecycle

- A set of **components and connectors** communicating through interface
- A set of **architecture design decisions**
- Focus on set of **views and viewpoints**
- Developed according to **architectural styles**

# Why Software Architecture?

# Interesting Rather Relevant Definitions

*“The shared understanding that the expert developers have of the system design”*

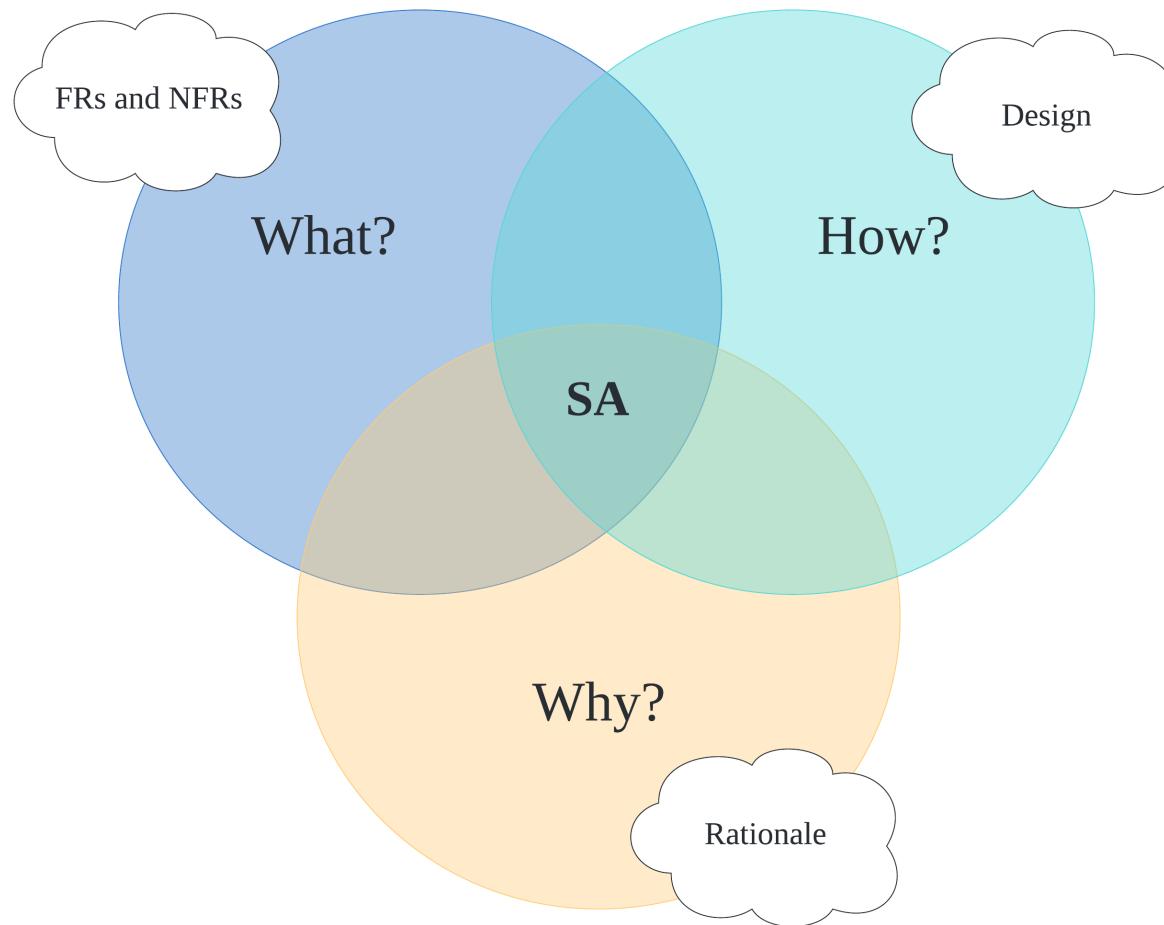
*“The decisions you wish you could get right early in a project”*

*“Software Architecture is about **all the important stuff**, whatever that is”*

Martin Fowler, *Who needs an Architect?* IEEE Software, 2003



# Abstraction



Manage complexity in the design (Transferable abstraction, promotes reuse)

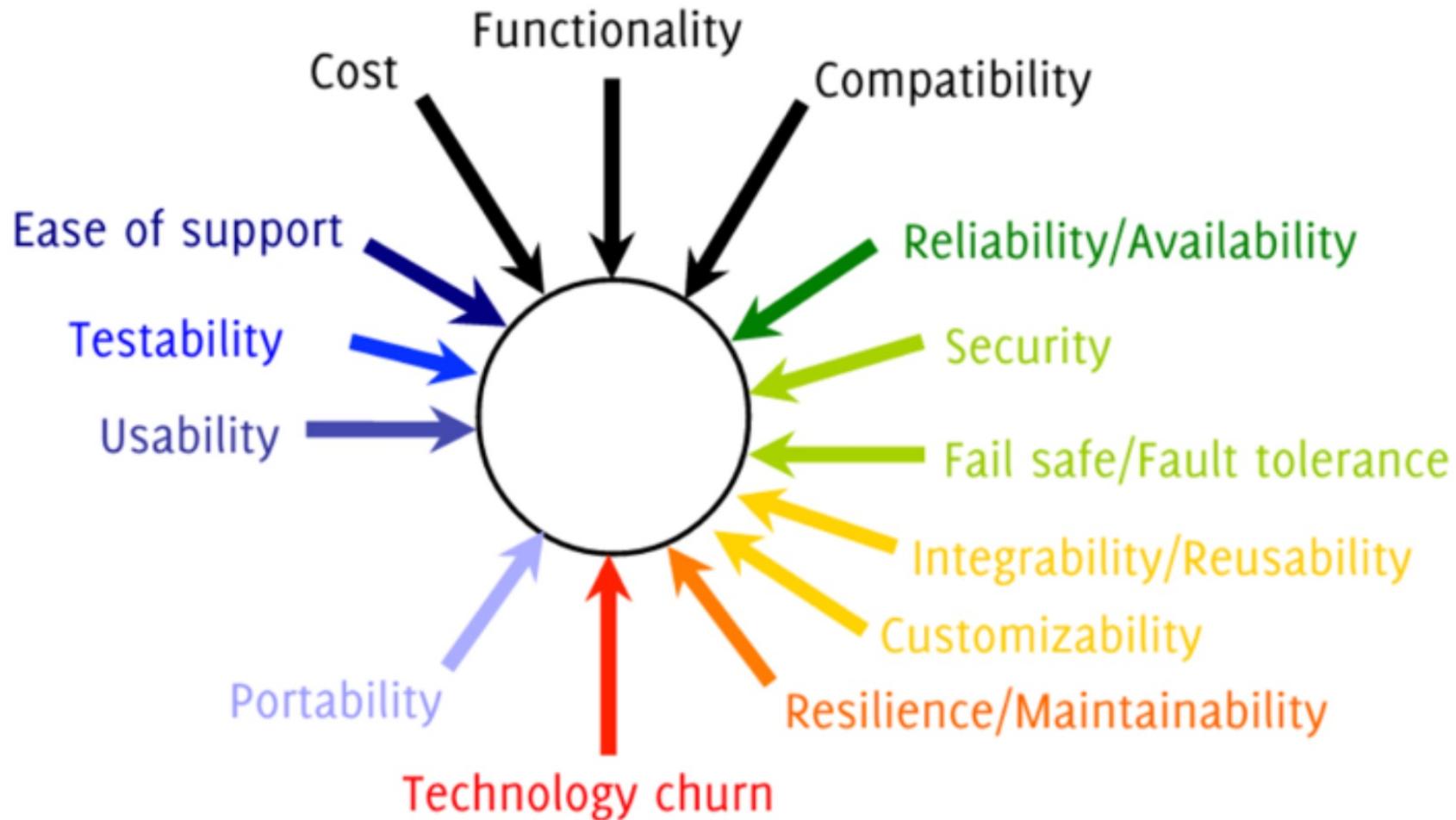
# Communication

*“Good architects are grown not born”*



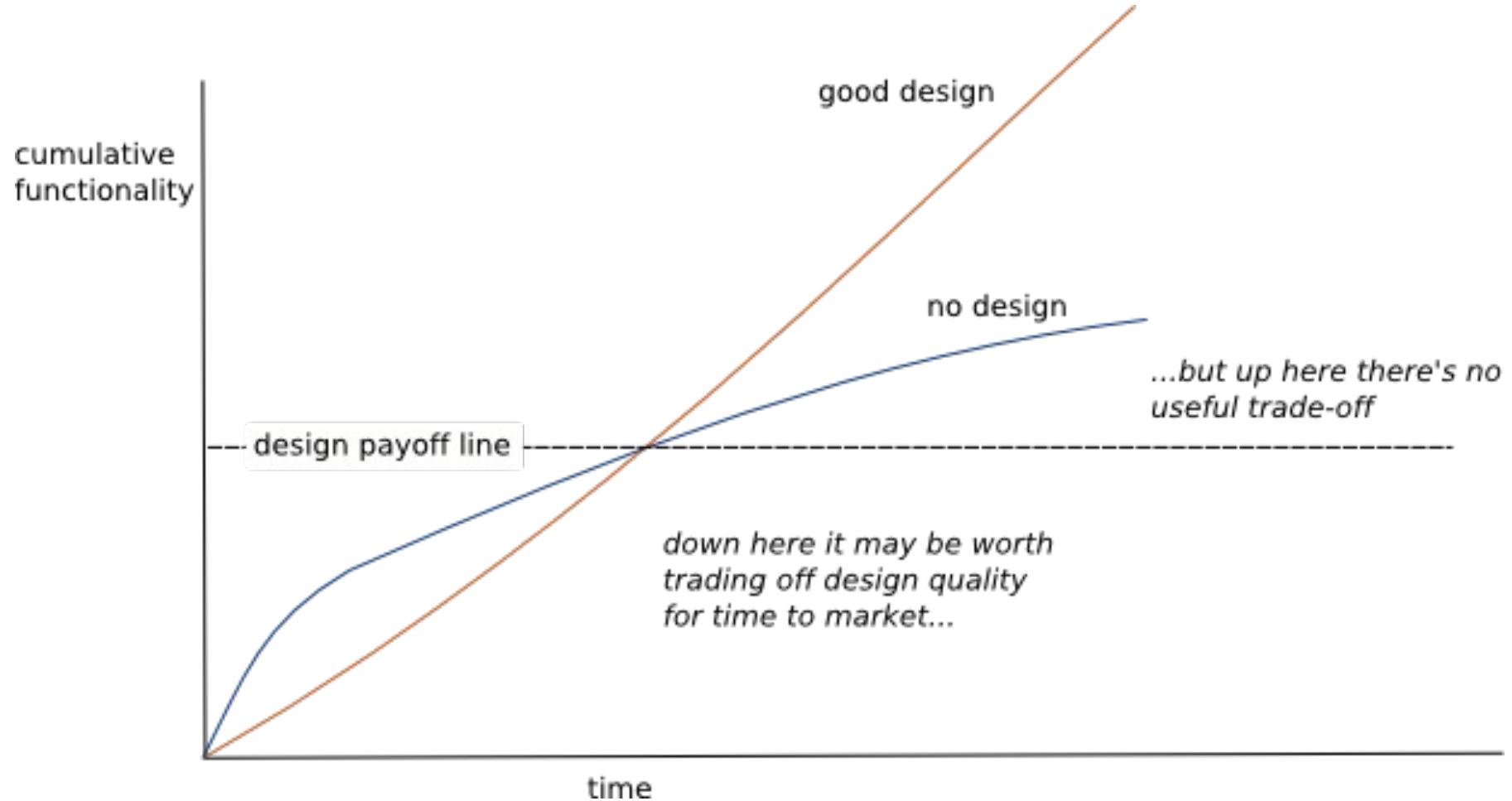
1. Document, remember and communicate among stakeholders
2. Manifest early design decisions

# Quality Analysis



Understand, Predict and Control

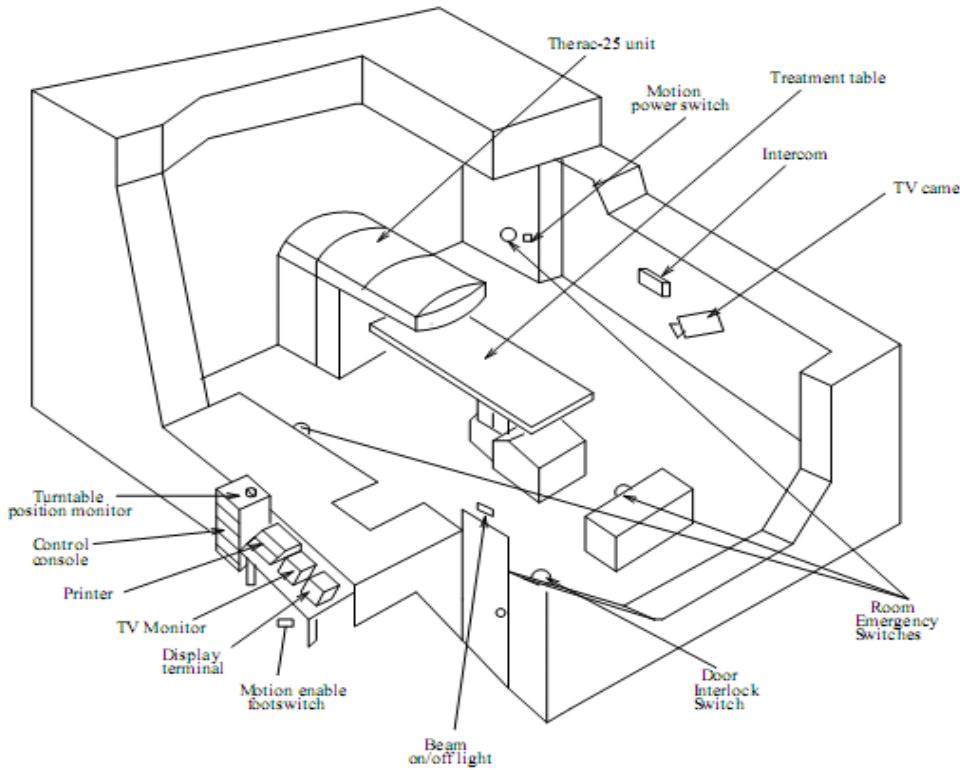
# Design Stamina Hypothesis



<https://martinfowler.com/bliki/DesignStaminaHypothesis.html>

# Why to Care? - The Case of Therac 25

- approximately 100 times the intended dose of radiation
- 3 people died, and 6 got injured



1. **Overconfidence in software!!**
2. In adequate software engineering practices
3. Design failure – No defensive approach (0 error handling or verification)

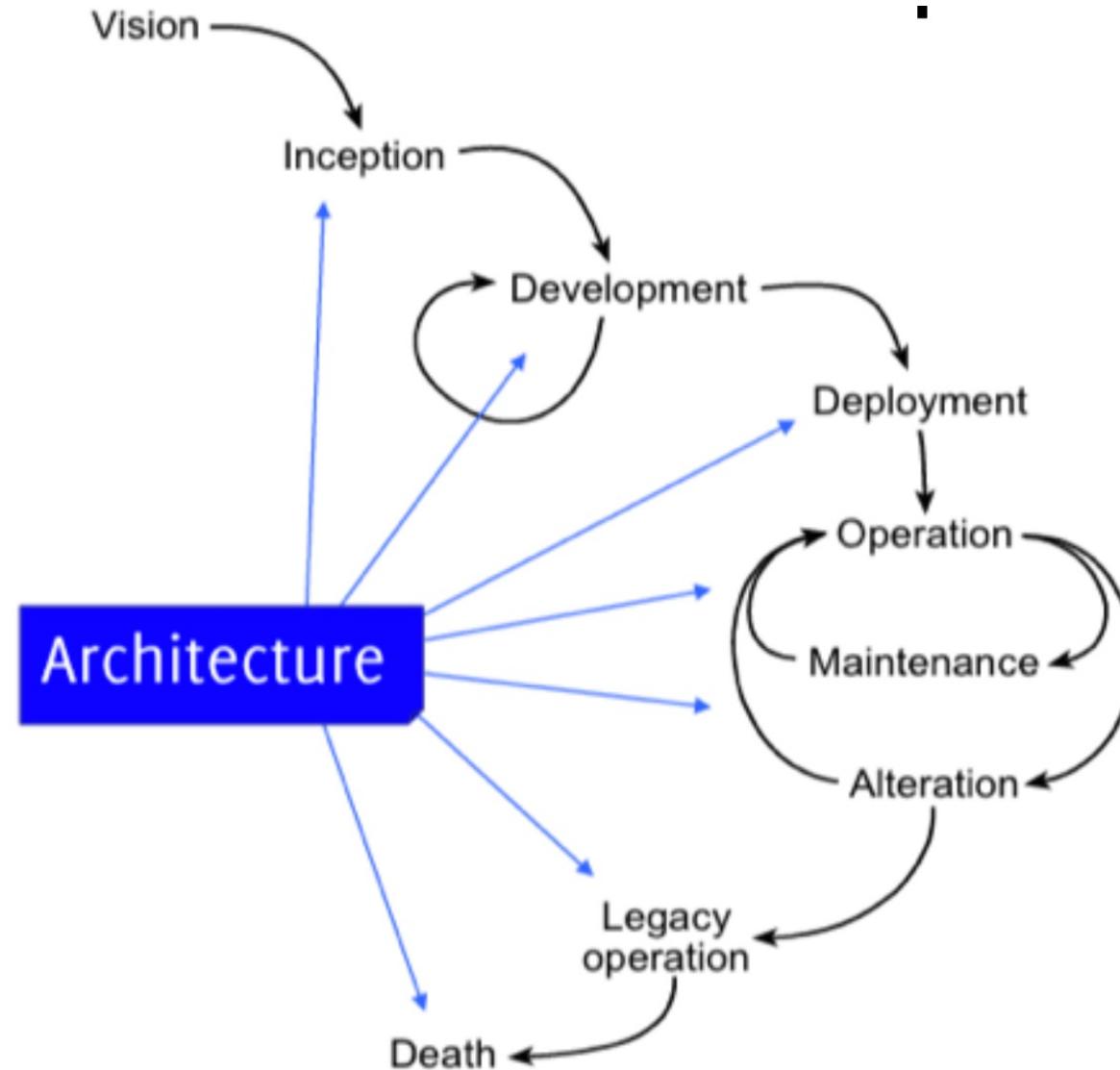
For more info: <http://sunnyday.mit.edu/papers/therac.pdf>

<https://www.computer.org/csdl/magazine/co/2017/11/mco2017110008/13rRUXASTVR>

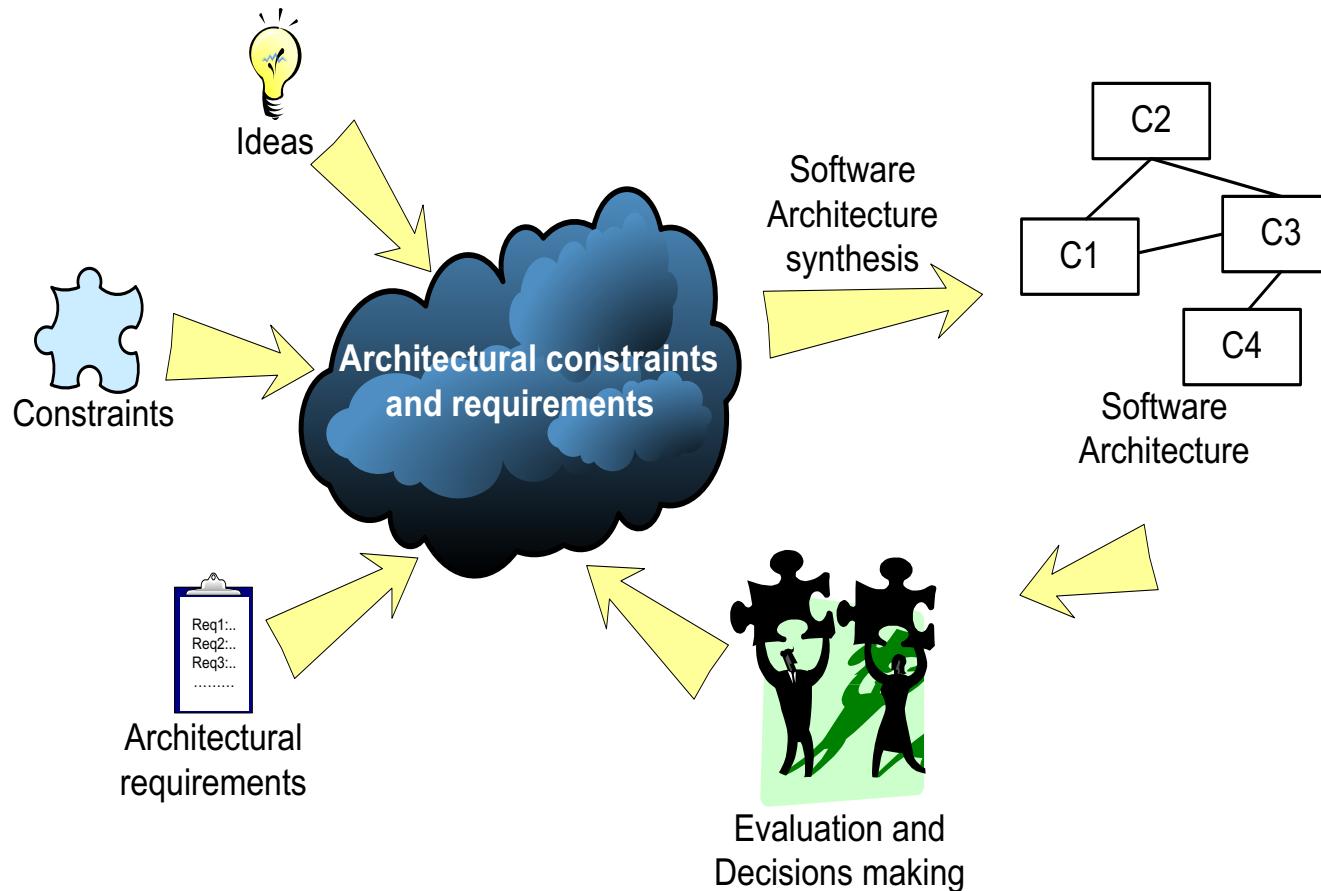
# Why to Care?

- All the software systems have an architecture
  - All the **critical/complex** systems must have it carefully and explicitly specified
- **Architecture-level decisions impact the scalability, performance, testability, functioning of the produced system**
- Even if the code is perfectly written, a wrong architecture produces a wrong system

# When does SA end?

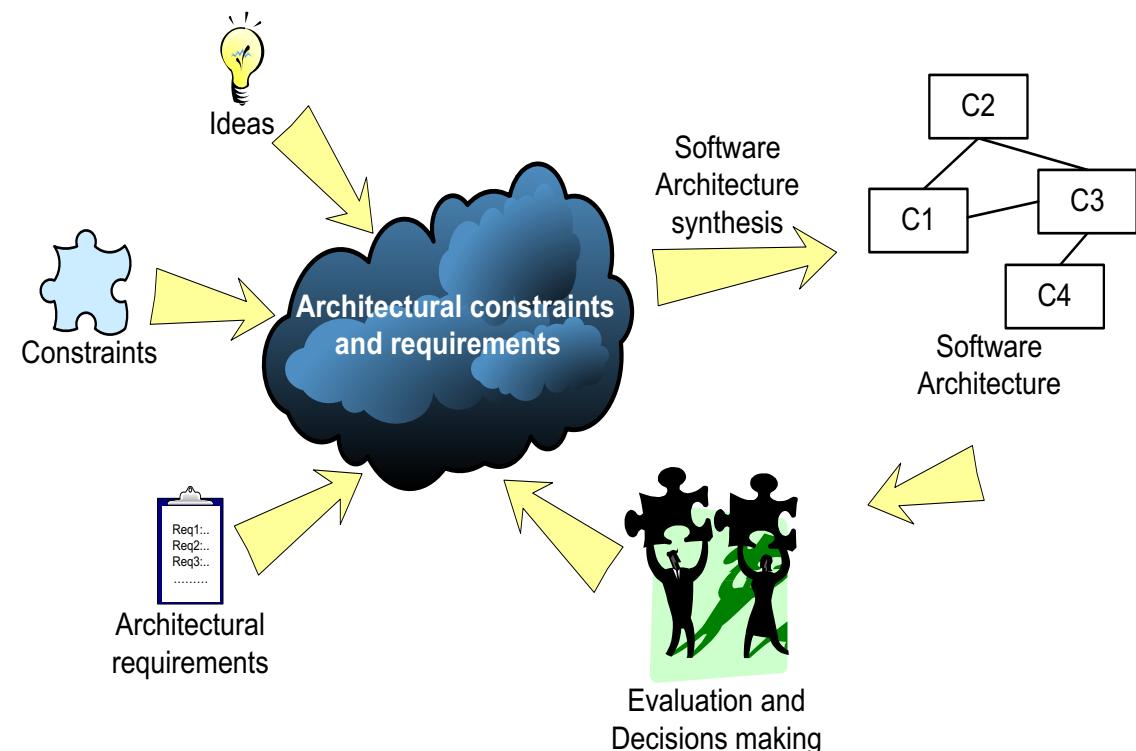


# The Overall Architecting Process



# So which is the right architecture?

- There is not one that is the best – tradeoff's
- The one that satisfies at best the requirements and constraints
- The "less" risky one



# Concrete Example

# The Case of Uffizi Gallery

- 3rd most visited museum in Italy in 2018
- More than 2.200.000 visitors per year
- Limited contemporary access for safety reasons
- Waiting time went sometime up to **4 hours!!**



**Goal:** Build a crowd management system

# Requirements for the System

## Functional Requirements:

1. FR1: User Registration
2. FR2: Check Availability
3. FR3: Entry booking
4. FR4: Recommendations

.....

## Non-Functional or Extra Functional Requirements:

1. EFR1: Performance – Latency/request < 0.1 sec
2. EFR2: Security – Prevent unauthorized access
3. EFR3: Availability – 99.999%
4. EFR4: Scalability – 1000 users/second
5. ...

Let the key requirements drive the high-level design of the system!!!

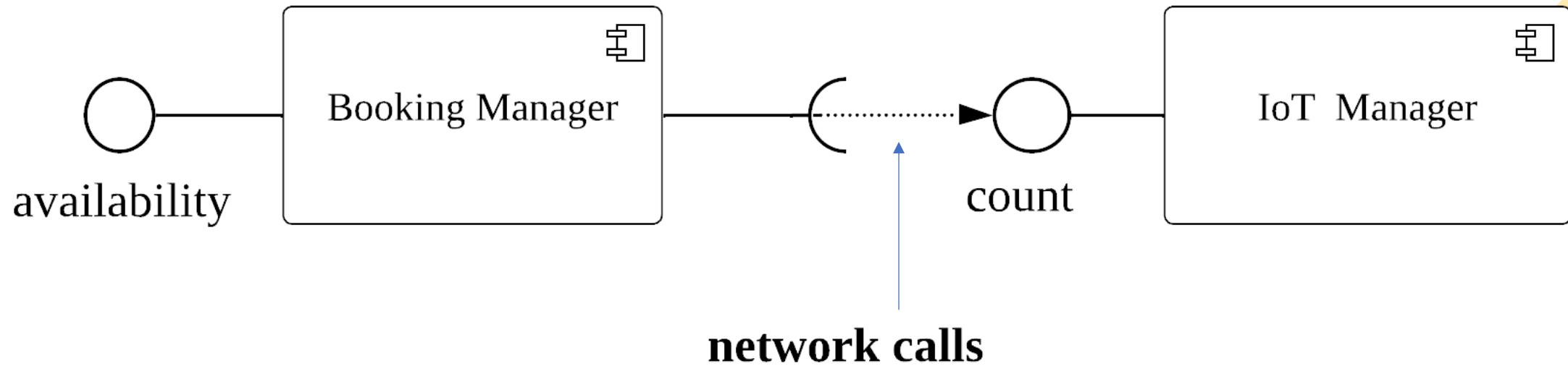
# Software Architecture

The Software Architecture is the earliest model of the whole software system created along the software lifecycle

- A set of **components and connectors** communicating through interface
- A set of **architecture design decisions**
- Focus on set of **views and viewpoints**
- Developed according to **architectural styles**

# Components and Connectors

# Components and Connectors



## Components:

- Data or processing element
- Has a **provided** and **required** interface

Eg: database, client, server, etc.

## Connectors:

- Enables interaction among components
- Can be implicit or explicit

Eg: HTTP events, procedure calls, etc.

# Design Decisions

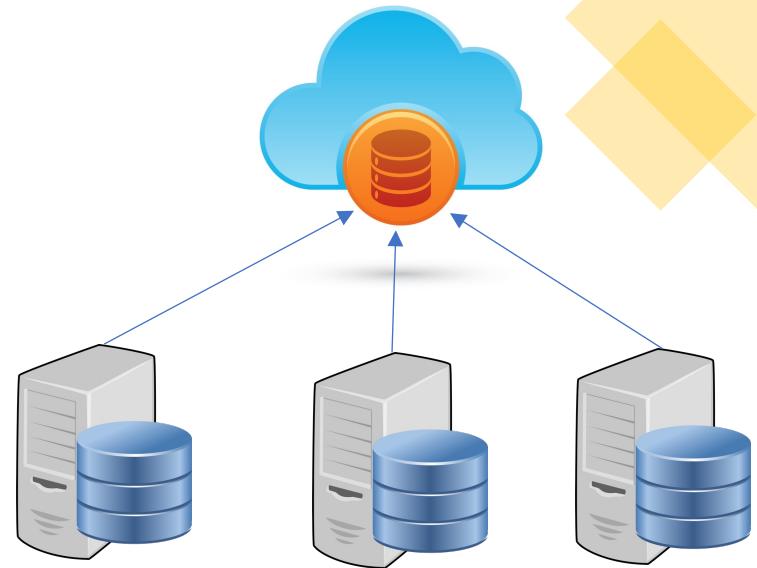
# Let us revisit our case – What to Choose?



Centralize data in cloud



Store data in Fog



Each museum can have its own data

Implications on performance, privacy, security, etc.

# Reasoning with Simple Logic may not work!

- Oracle is more scalable than MySQL
- MySQL is more scalable than Informix

Therefore Oracle is more scalable than Informix

**Q:** I need a scalable RDBMS, Shall I got with Oracle?

**A:** It depends!!!



# Architectural Design Decisions

Decisions about:

- Selected components/interfaces/connectors
- Distribution/Configuration of components/connectors
- Expected behavior
- SA Styles, Patterns and Tactics
- HW/SW/Deployment and other views
- Components' Nesting and sub-systems
- NF attributes

# Consequences of Design Decisions

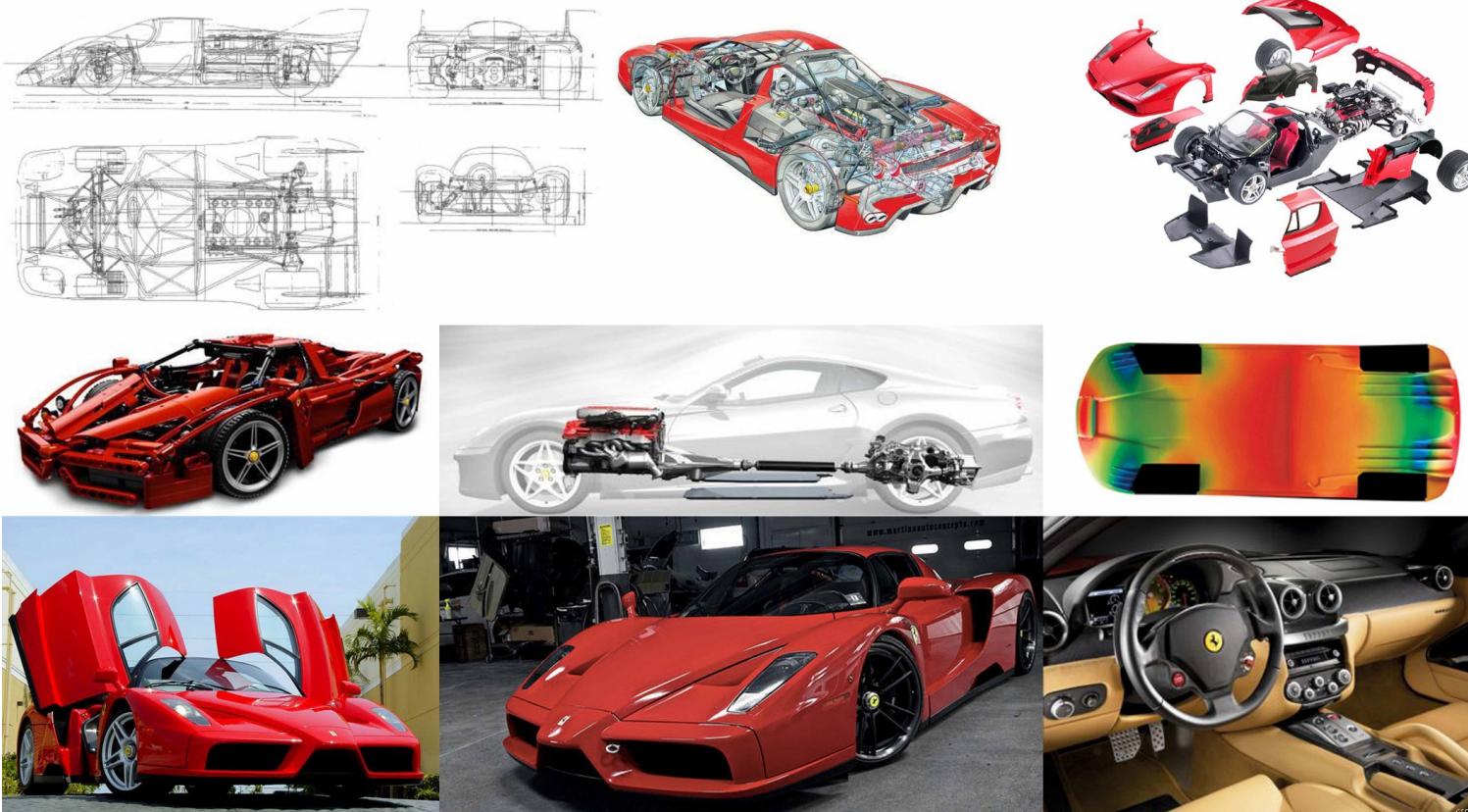
- Defines constraints on implementation
- Dictates organizational structure
- Inhibits or enables system's quality attribute
- System qualities may be predicted
- Easier to manage change
- Helps in evolutionary prototyping
- Enables cost and schedule estimates



# Views and Viewpoints

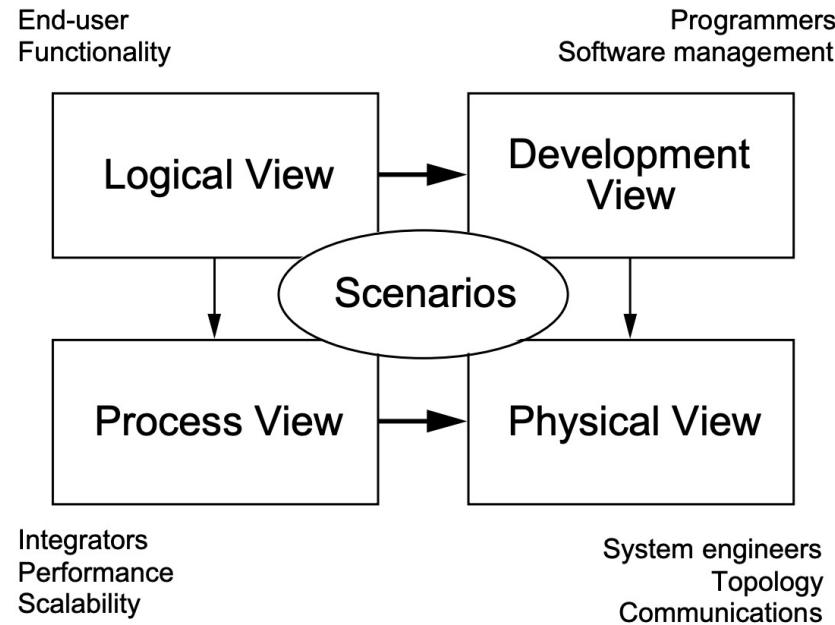
# Architecture View and Viewpoints

- Viewpoint is about where you see from
- View is what you see!! – Viewpoint governs the view

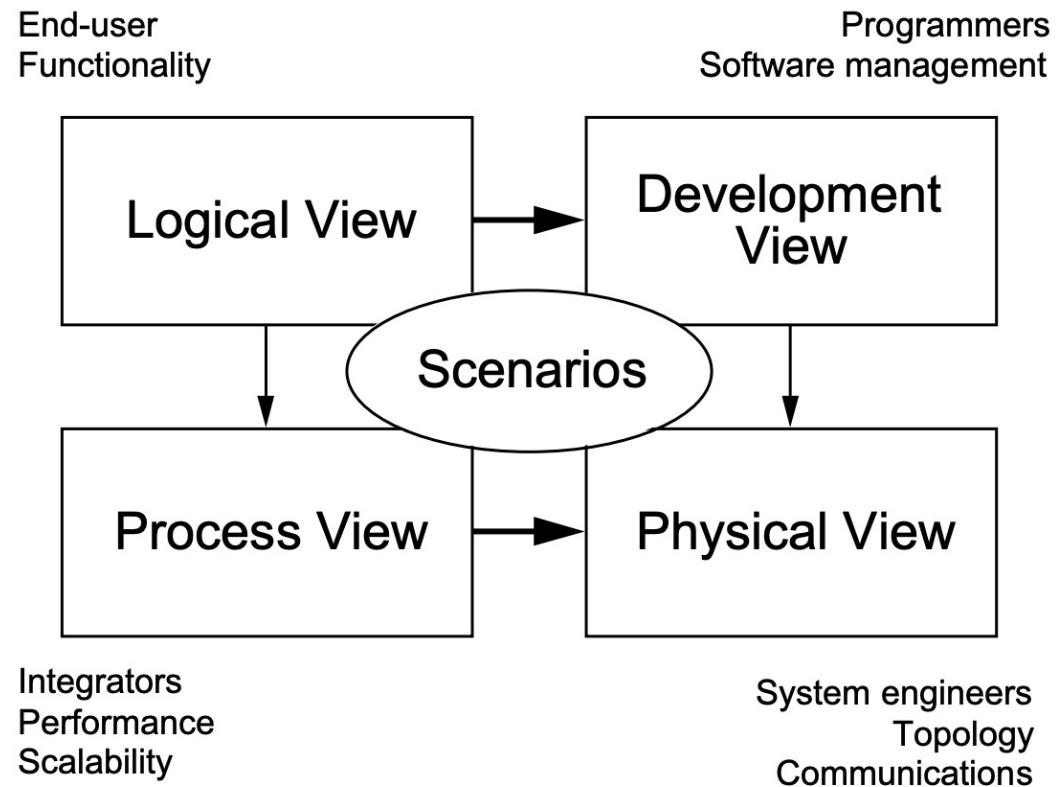


# Architectural Views – How Many?

- View represents a collection of architectural elements and relations among them
- Two fundamental views – Structural and Behavioral
- Many models have been proposed – eg: 4+1 view model



# 4+1 View Model of Software Architecture

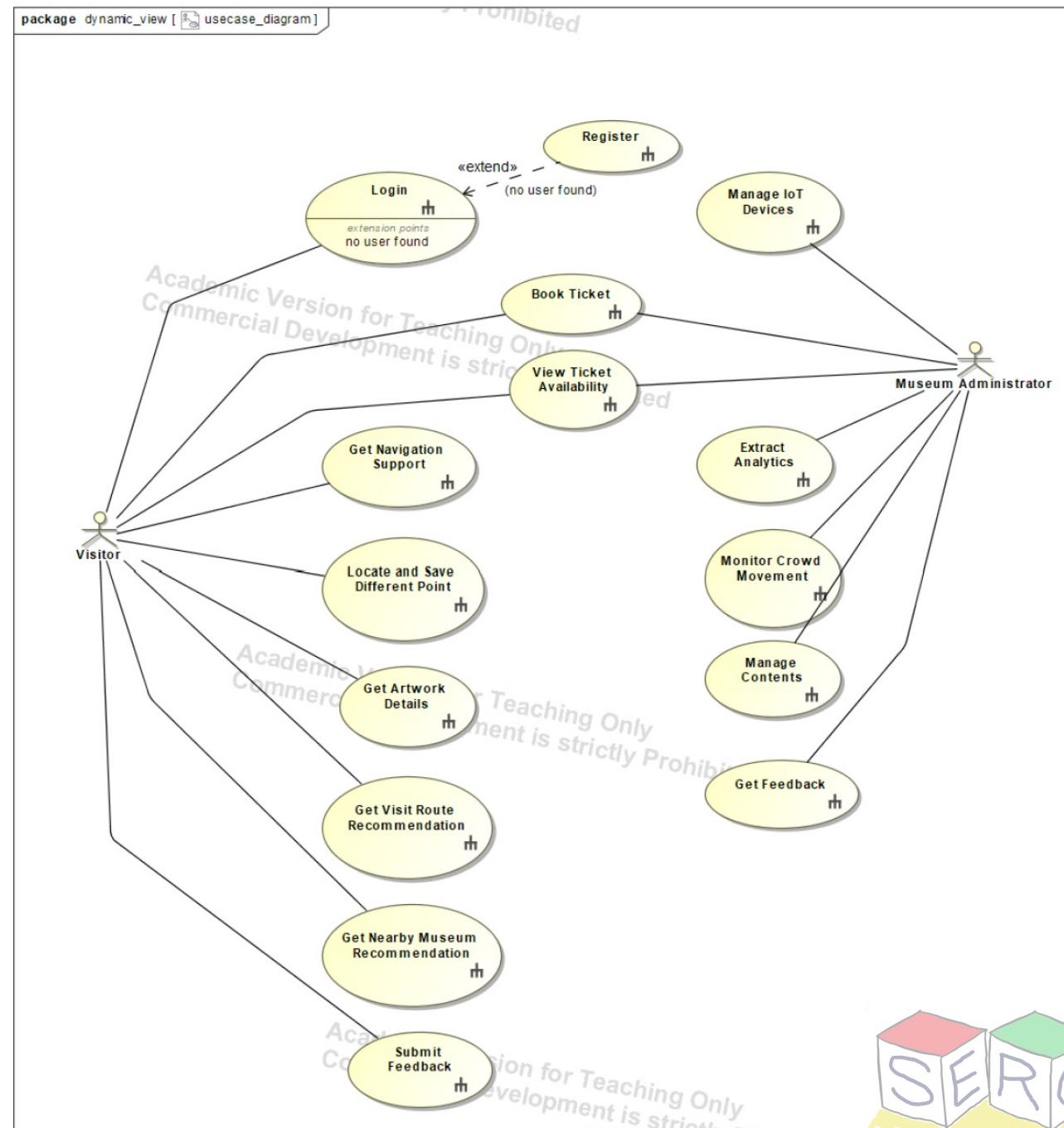


*The “4+1” view model is rather “generic”: other notations and tools can be used, other design methods can be used, especially for the logical and process decompositions, but we have indicated the ones we have used with success.*

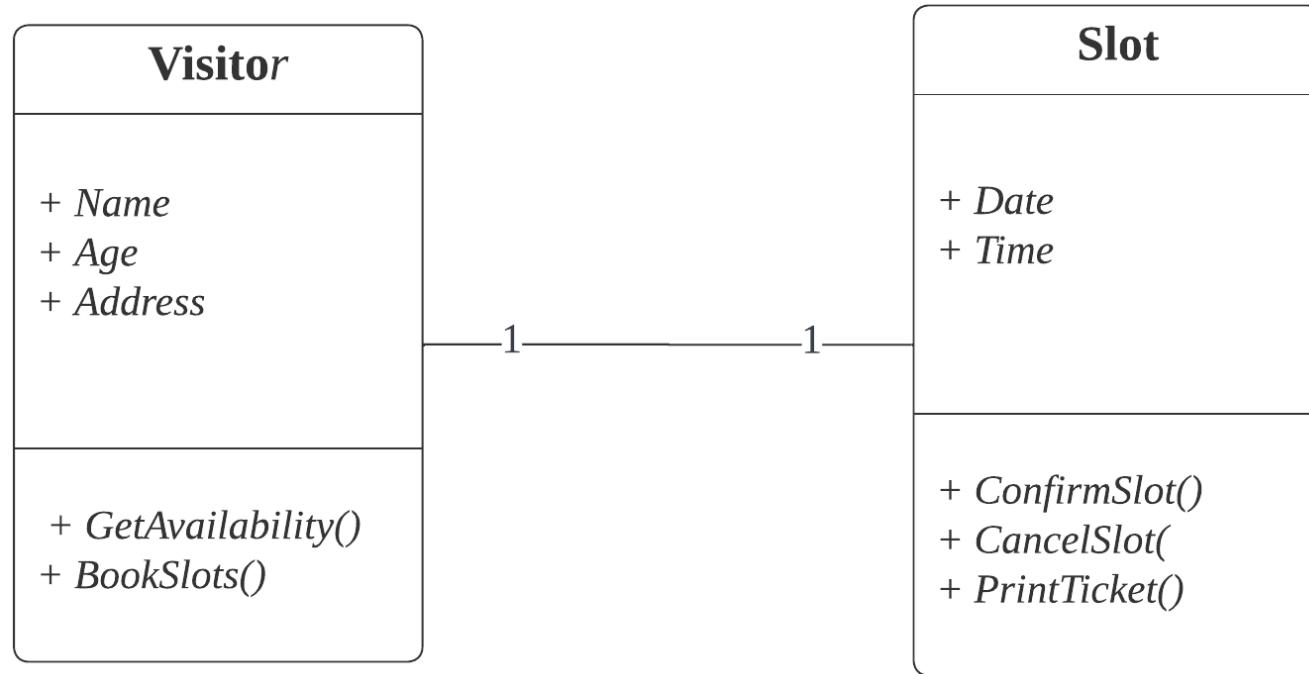
- Philippe Kruchten, Architectural Blueprints—The “4+1” View Model of Software Architecture

# Scenarios

- Represent the different use cases
- **Stakeholders:** End-user, developer
- **Concerns:** Understandability
- **Diagram:** Use case diagrams



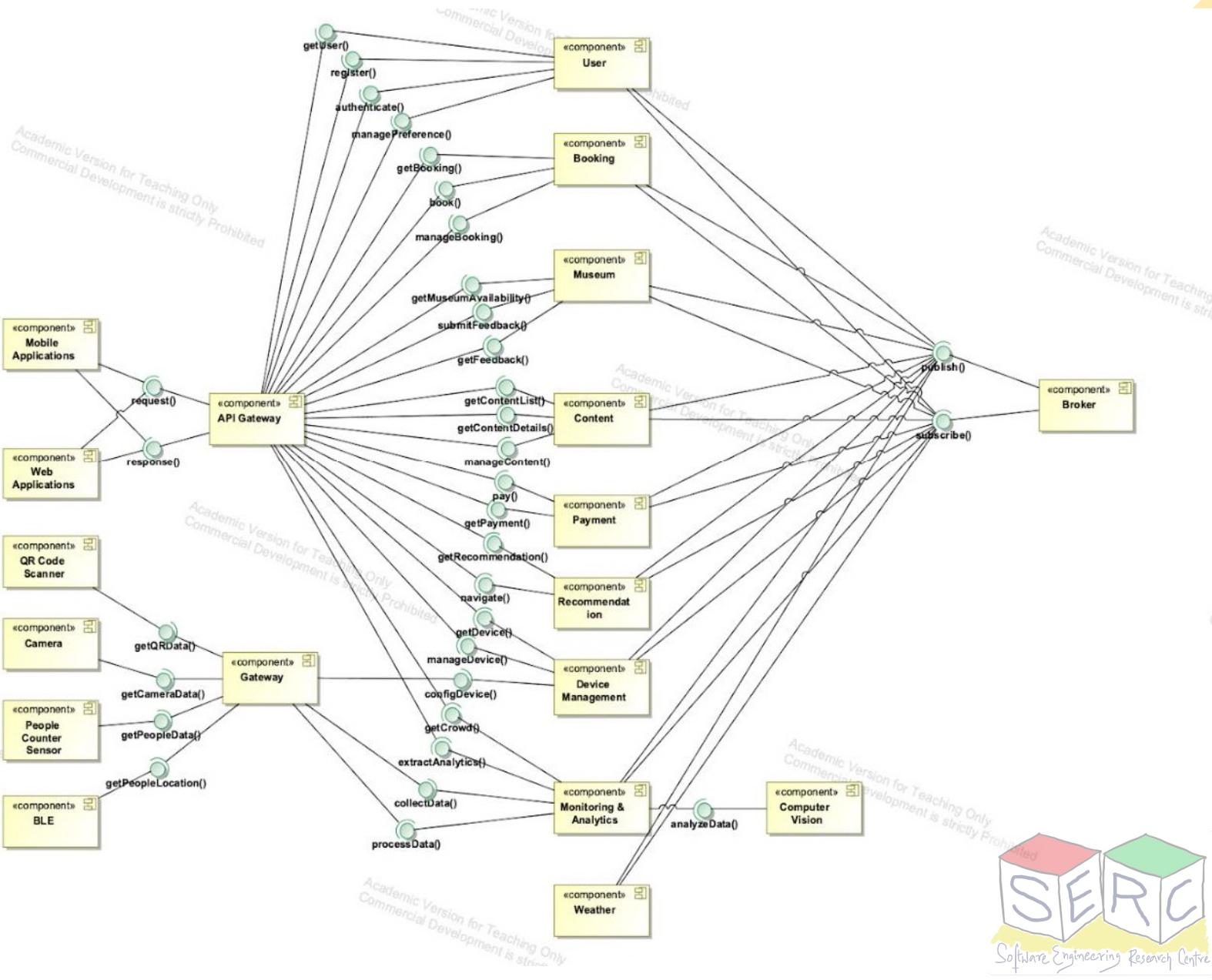
# Logical View



- System decomposed into a set of abstractions (objects or object classes)
- **Stakeholders:** Developer
- **Concerns:** Functionality
- **Diagrams:** UML Class diagrams, logical connection diagrams

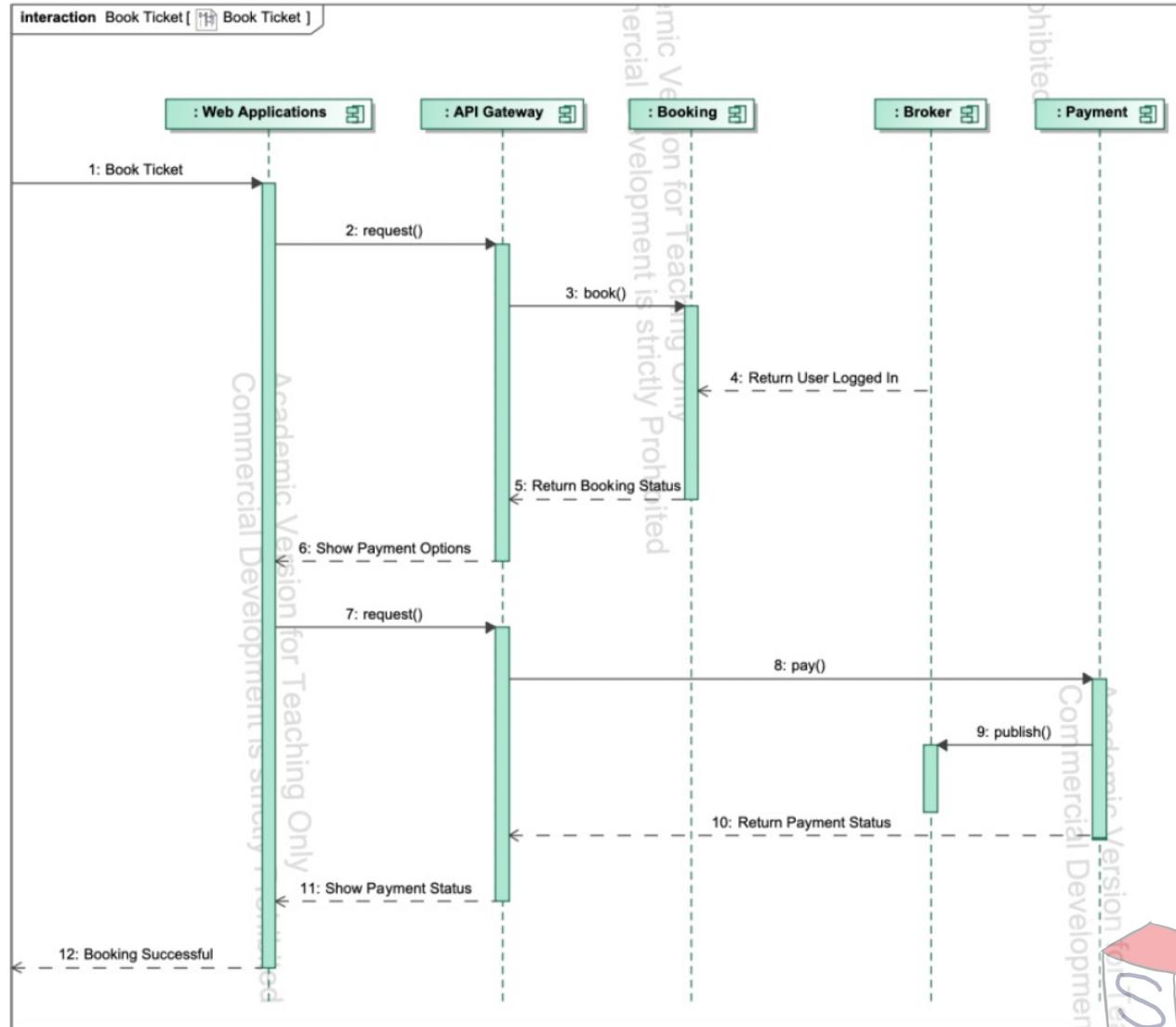
# Development View

- Organization of software into subsystems/modules
- Stakeholders: Developer, manager
- Concerns: Organization, reuse, portability
- Diagram: UML Component diagram



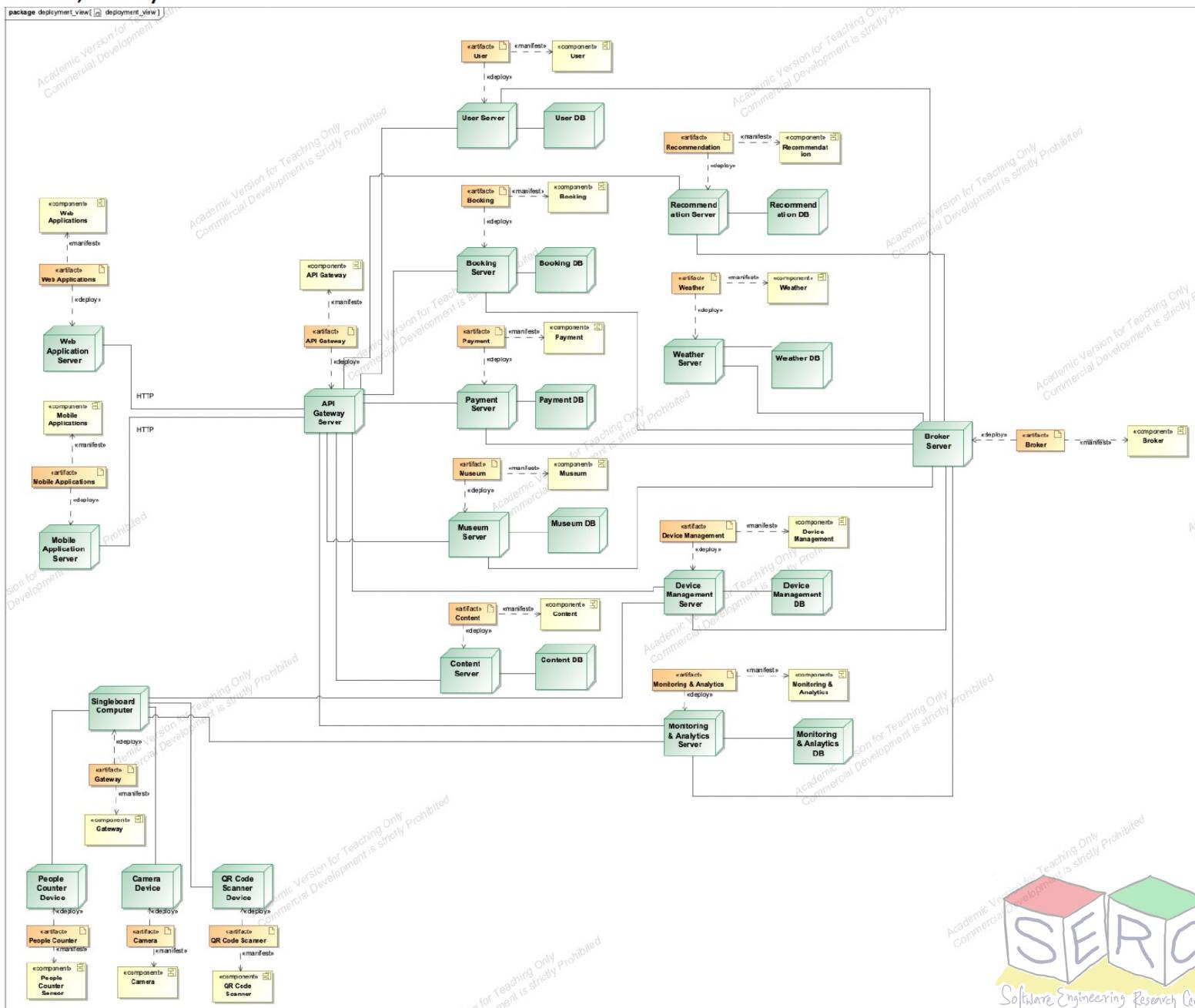
# Process View

- Model dynamic aspects of software architecture
- **Stakeholders:** System designer, integrator
- **Concerns:** Performance, fault tolerance
- **Diagram:** UML Sequence diagram, Process diagram, Data flow



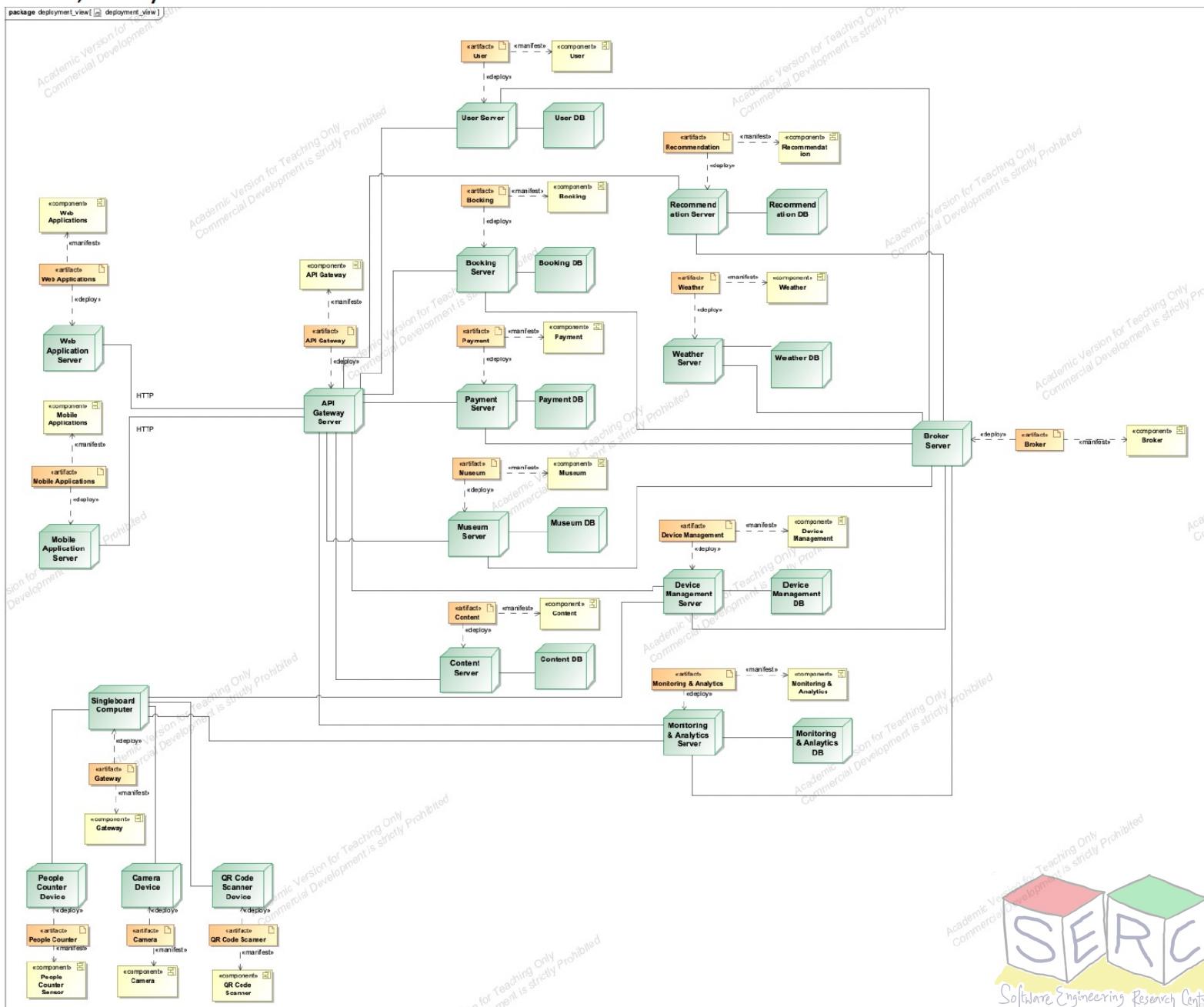
# Physical View

- Mapping of SW elements into deployment nodes
- **Stakeholders:** System designer, Admin
- **Concerns:** Performance, Scalability, Availability
- **Diagram:** UML Deployment diagram, Network diagram, etc.



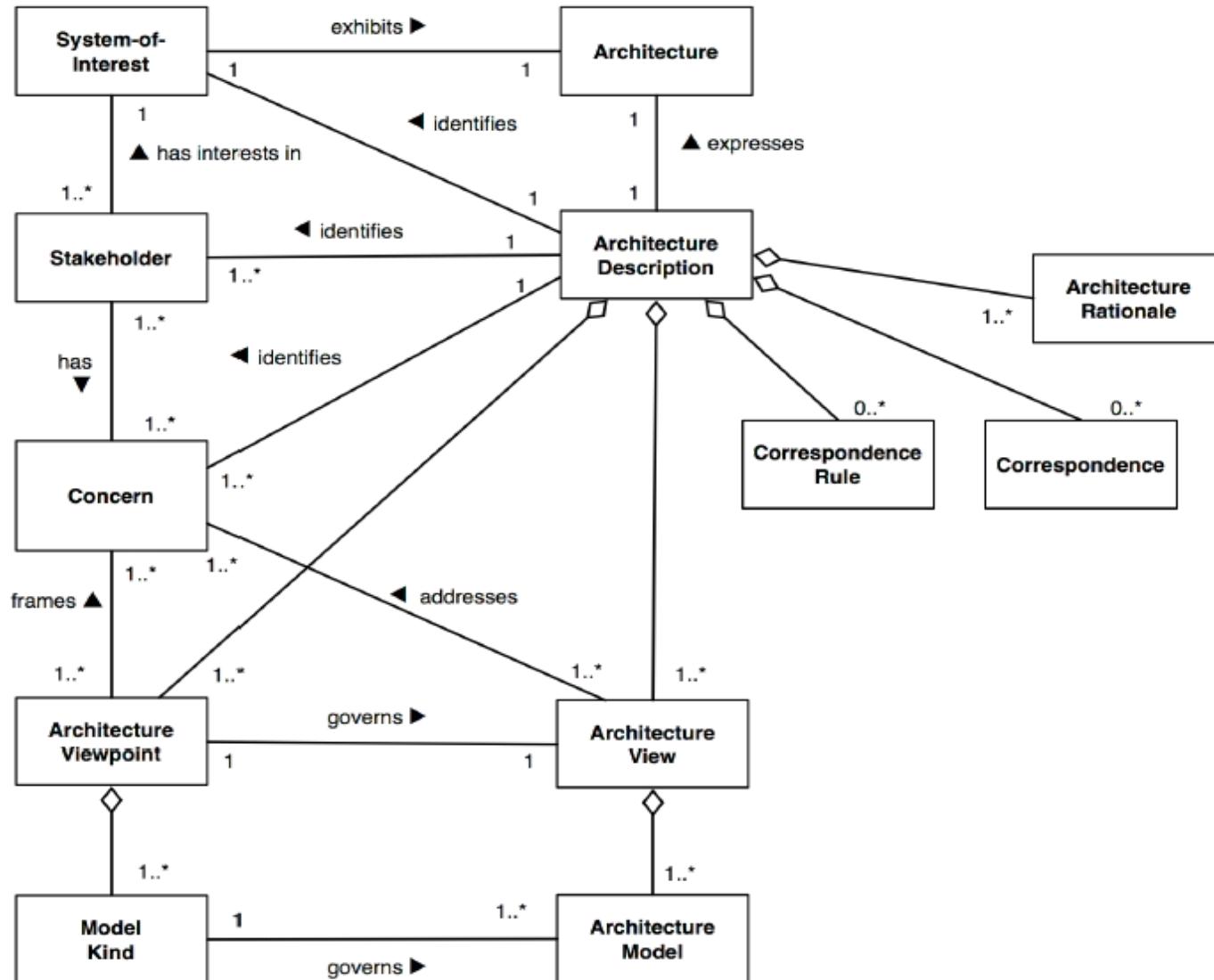
# Physical View

- Mapping of SW elements into deployment nodes
  - **Stakeholders:** System designer, Admin
  - **Concerns:** Performance, Scalability, Availability
  - **Diagram:** UML Deployment diagram, Network diagram, etc.



# Architecture Description

# Architecture Description



# Thank You



Course website: [karthikv1392.github.io/cs6401\\_se](https://karthikvaidhyanathan.github.io/cs6401_se)

Email: [karthik.vaidhyanathan@iiit.ac.in](mailto:karthik.vaidhyanathan@iiit.ac.in)

Web: <https://karthikvaidhyanathan.com>

Twitter: [@karthyishere](https://twitter.com/karthyishere)

