Introduction to Software Architecture

CS6.401 Software Engineering

Dr. Karthik Vaidhyanthan

karthik.vaidhyanathan@iiit.ac.in

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

SERC

Software Engineering Research Centre

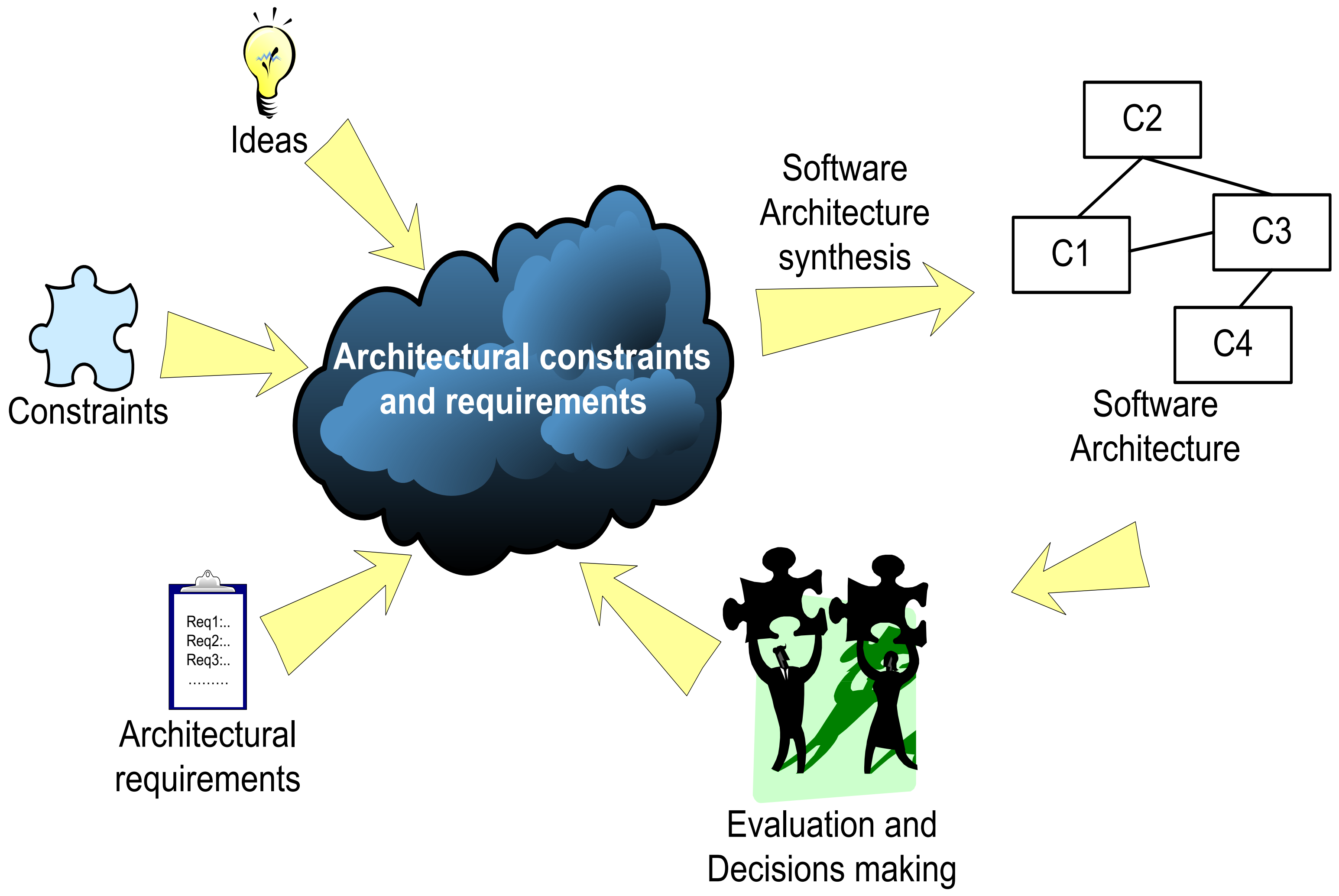
Acknowledgements

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

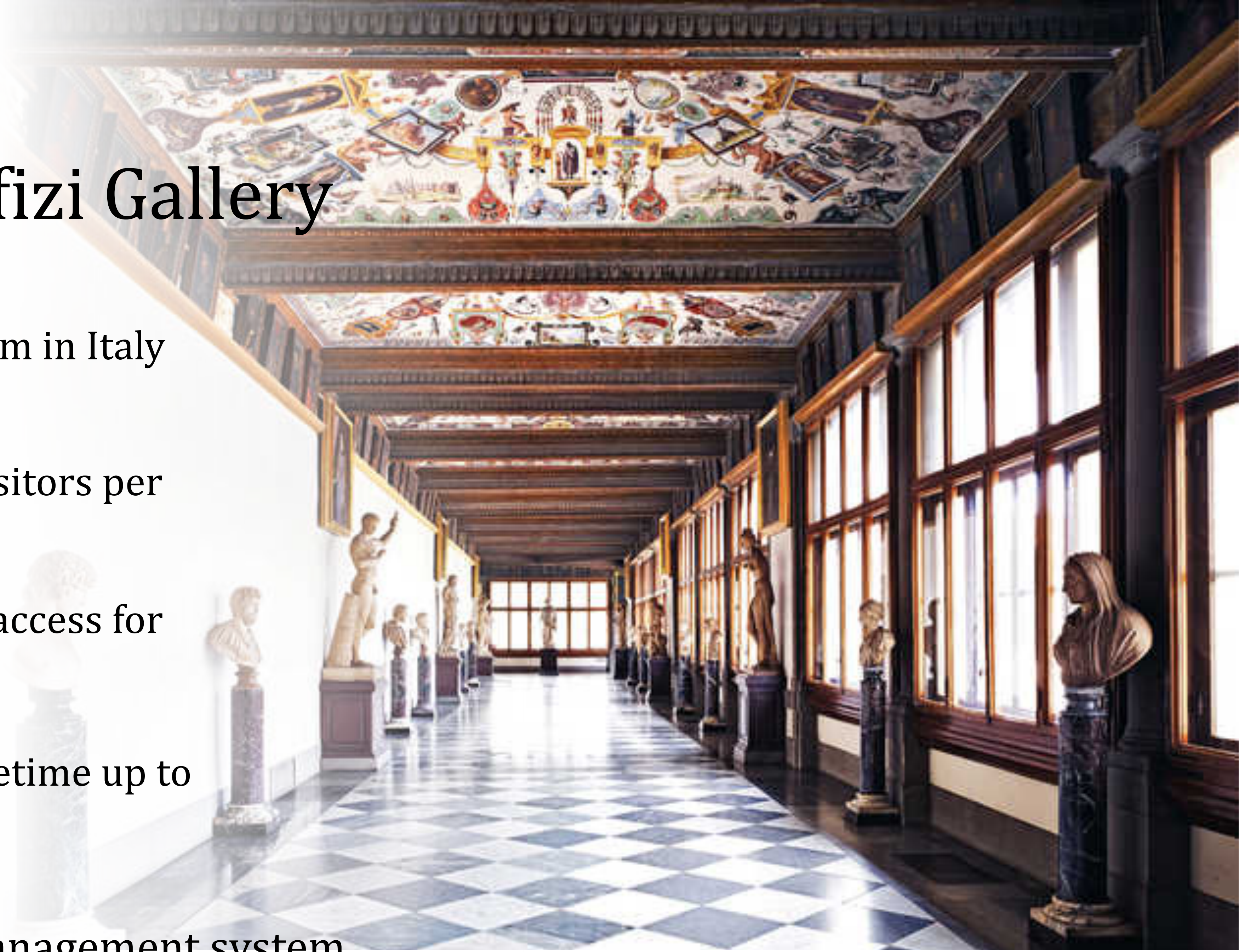
Sources:
1. Introduction to Software Architecture, Henry Muccini, University of L'Aquila
2. What is Software Architecture, Raghu Reddy, IIIT Hyderabad
3. Software Architecture in Practice, Len Bass, 3rd edition
4. Software Architecture (SE Course), Alessio Gambi, Saaraland University, Germany
5. Software Architecture Design Reasoning Workshop, Antony Tang, ISAPS 2018

Source: Googe images

The Overall Architecting Process


Ideas
Constraints
Architectural constraints and requirements
Software Architecture synthesis
C2
C1
C3
C4
Software Architecture
Req1:..
Req2:..
Req3:..
....
Architectural requirements
Evaluation and Decisions making

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

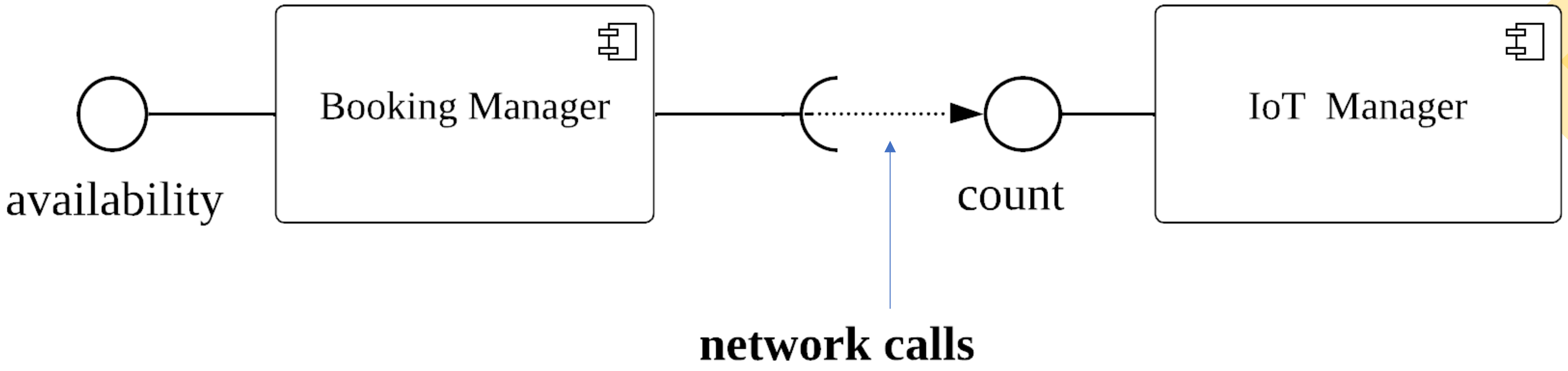
The Software Architecture is the **earliest model** of the **whole software system** created along the software lifecycle

- 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 explict

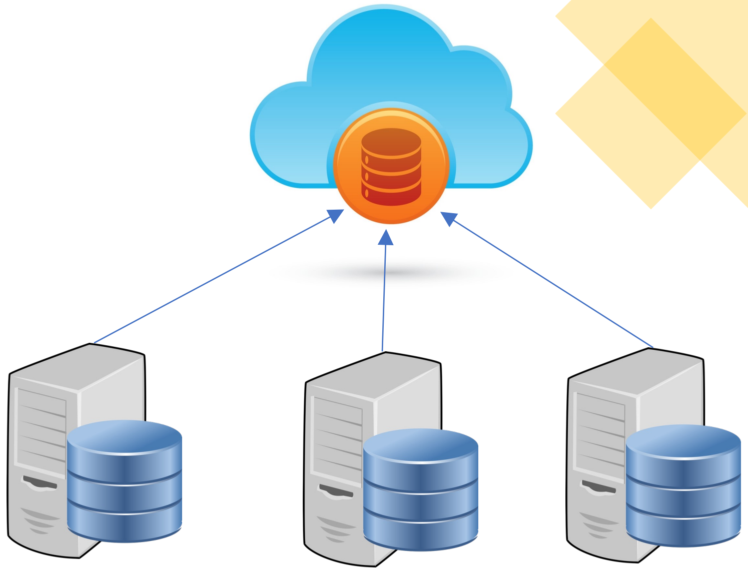
Eg: HTTP events, proceduce calls, etc.

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

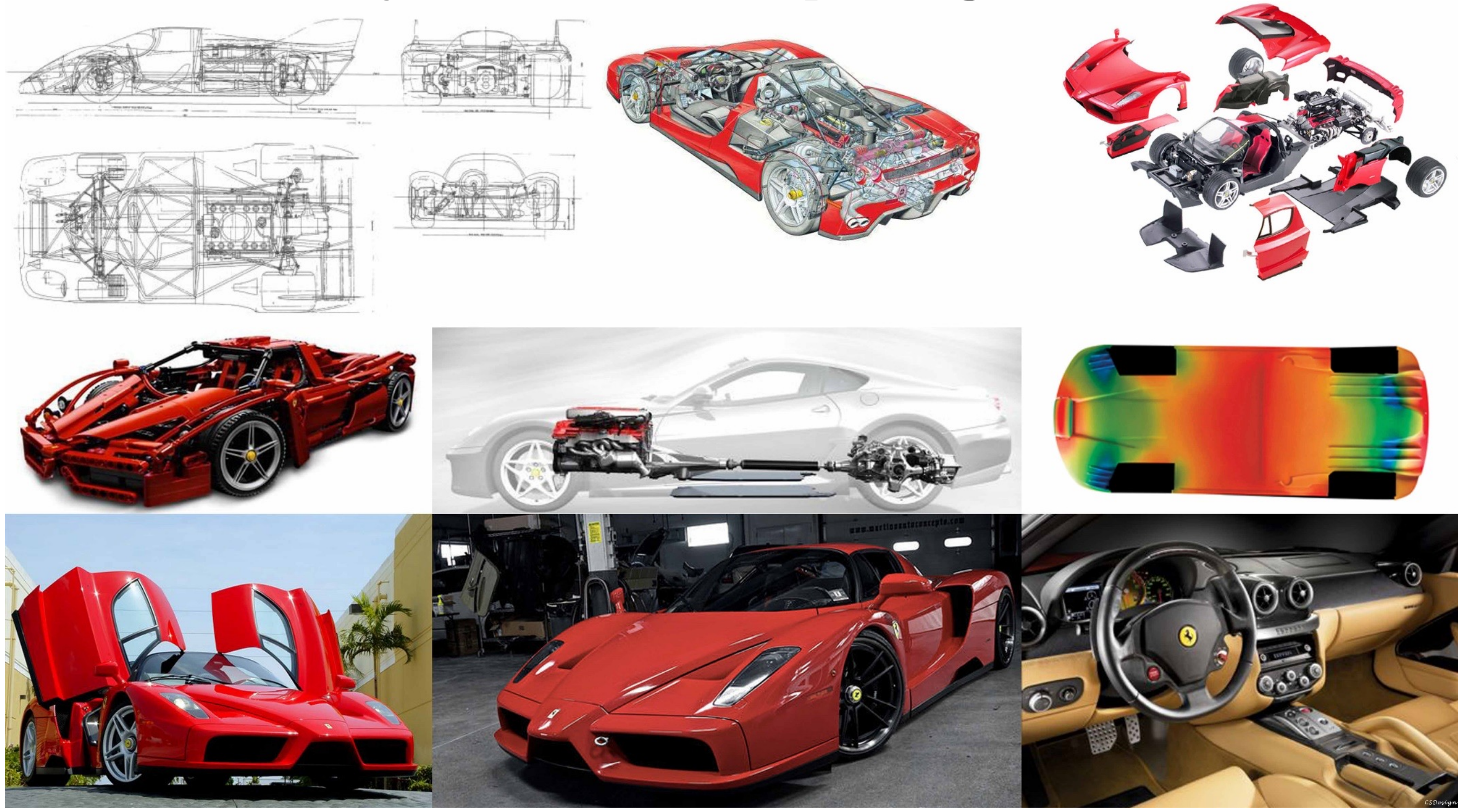
Consequences of Design Decisions

- 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

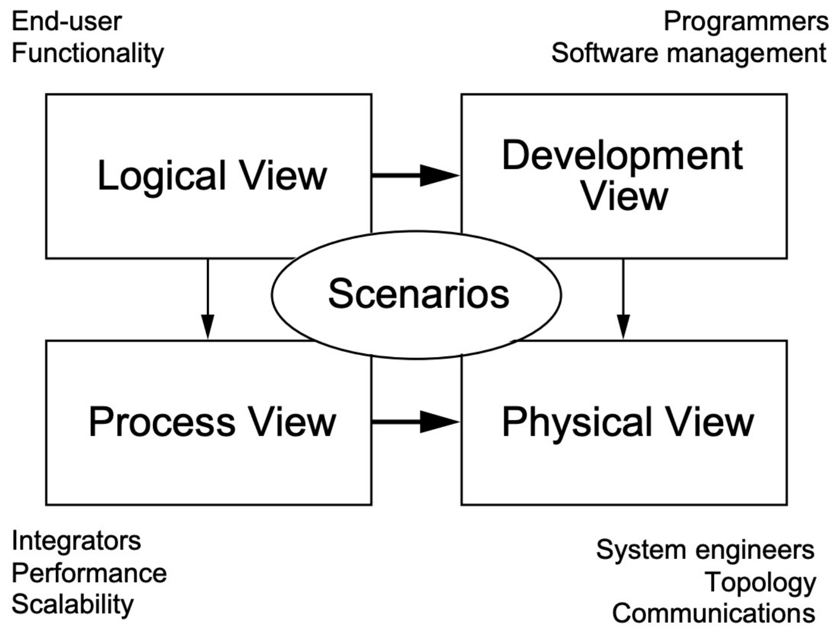
Architecture View and Viewpoints

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

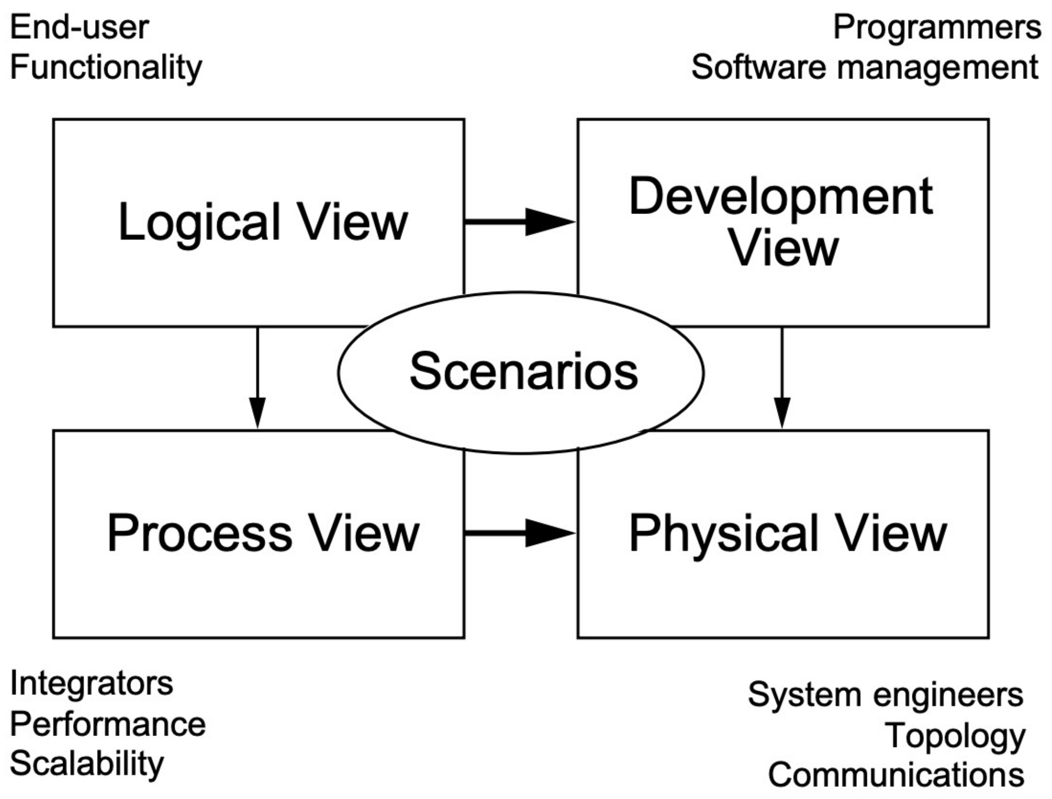


Architectural Views – How Many?

- 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

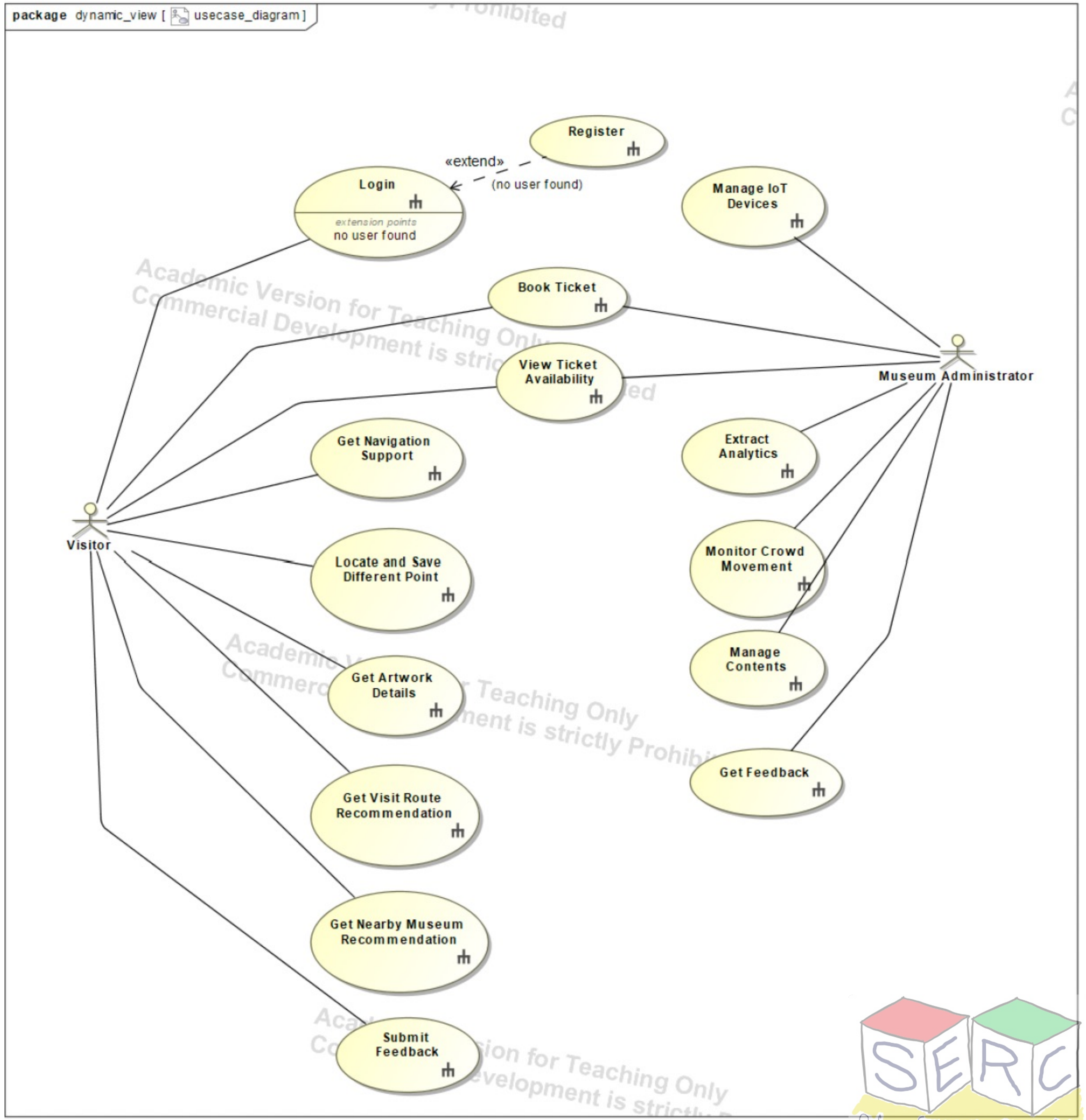


The "4+1" view model is rather "generic": other notations and tools can be used, other design methods can be used, especially for the logical and process decompositions, but we have indicated the ones we have used with success.

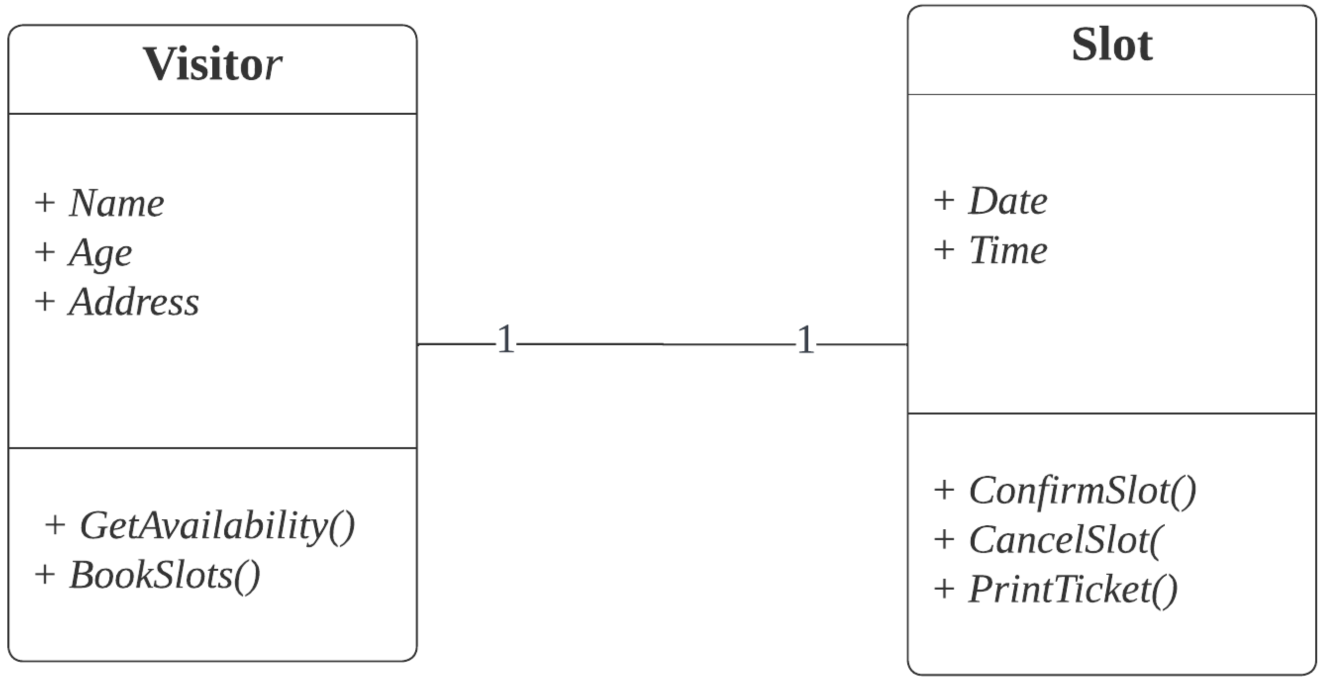
- Philippe Kruchten, Architectural Blueprints—The "4+1" View Model of Software Architecture

Scenarios

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



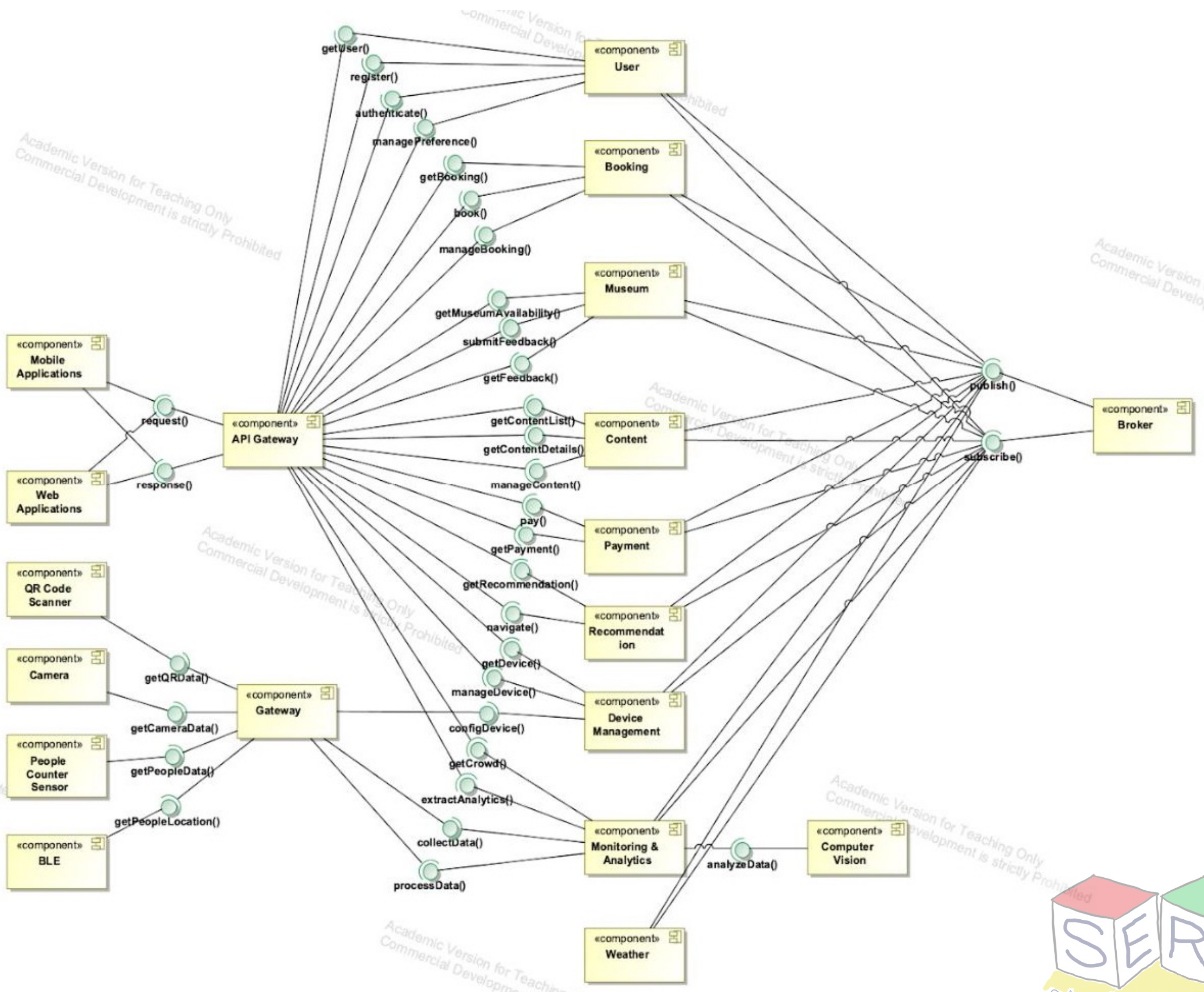
Logical View



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

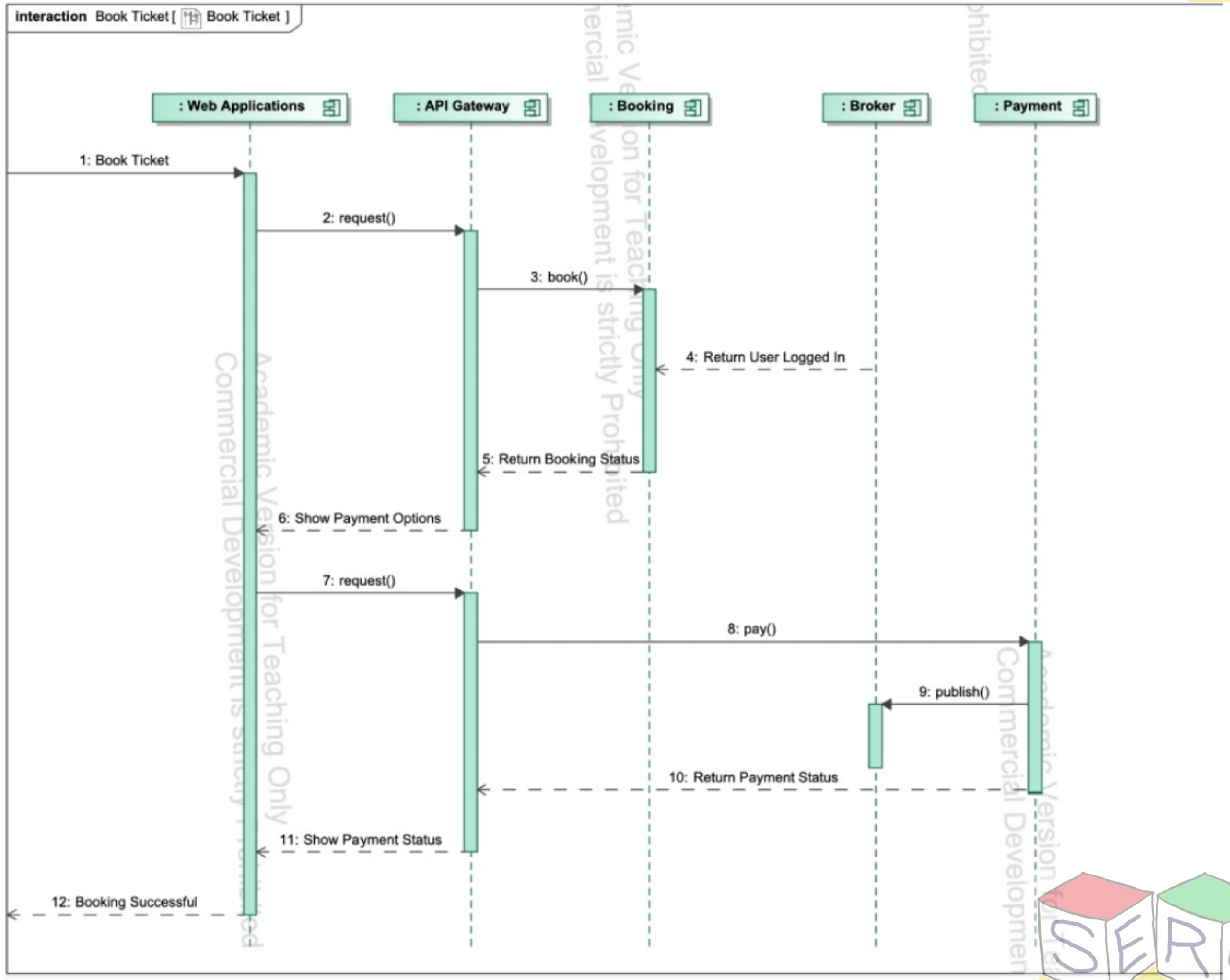
Development View

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



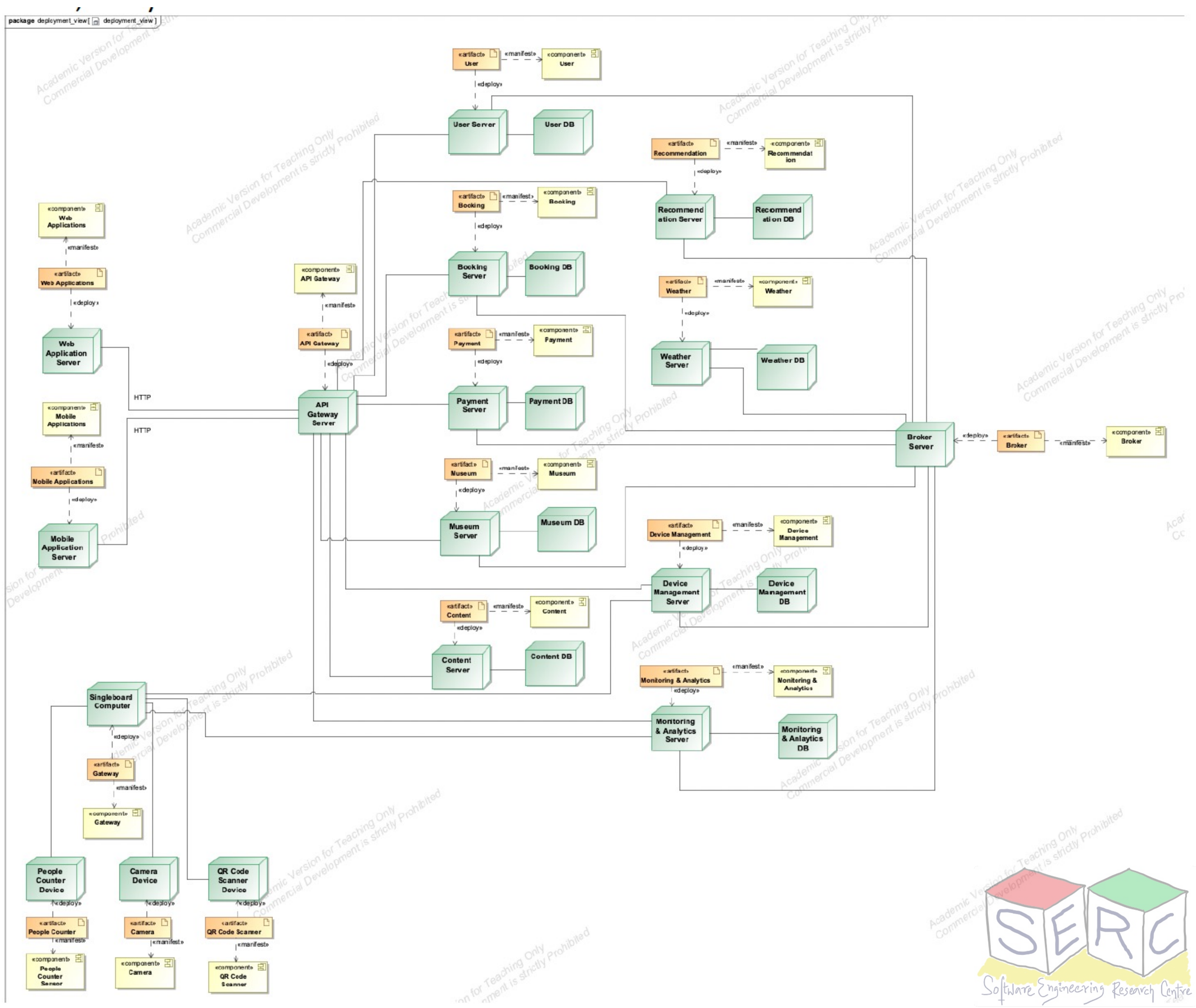
Process View

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



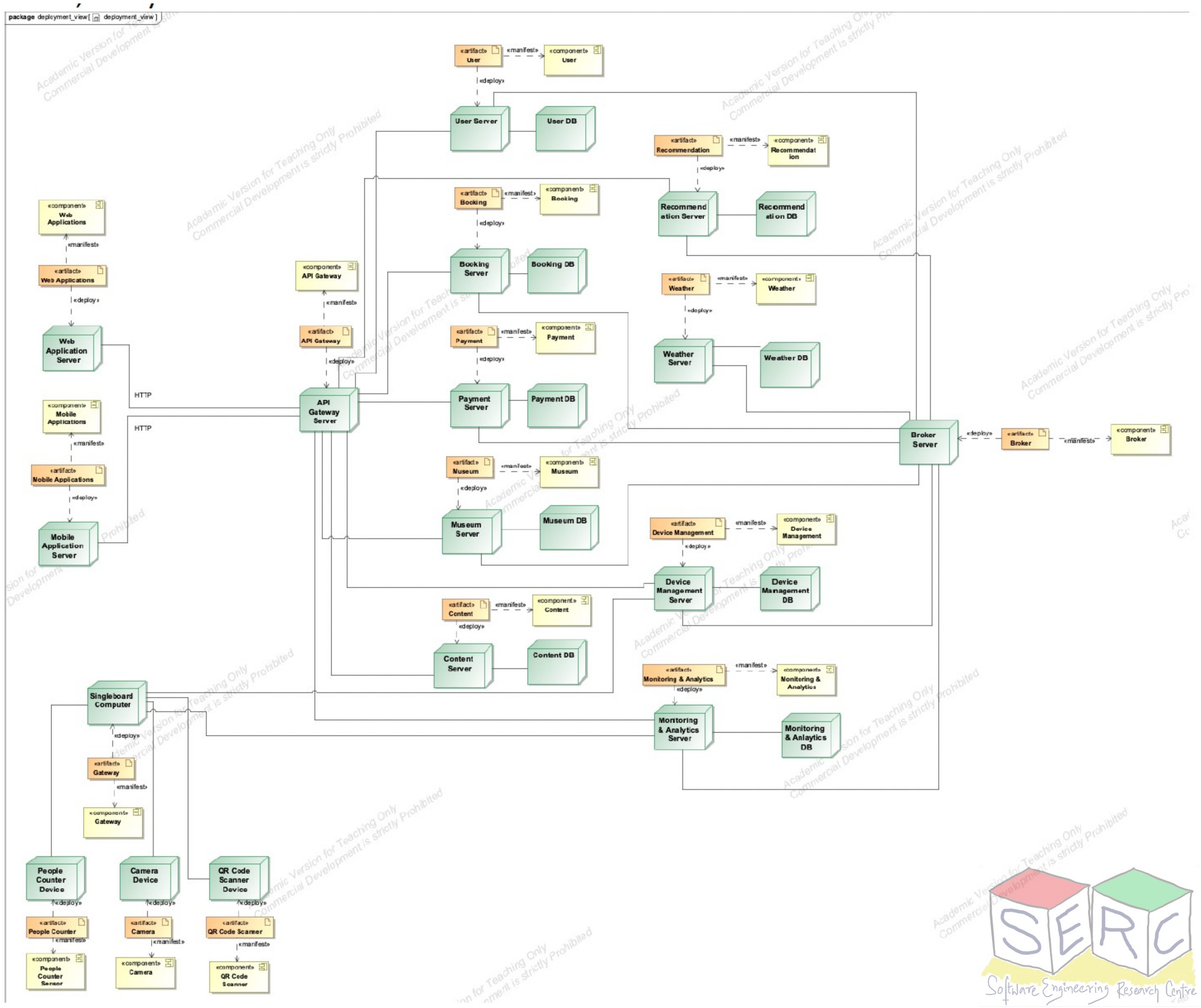
Physical View

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



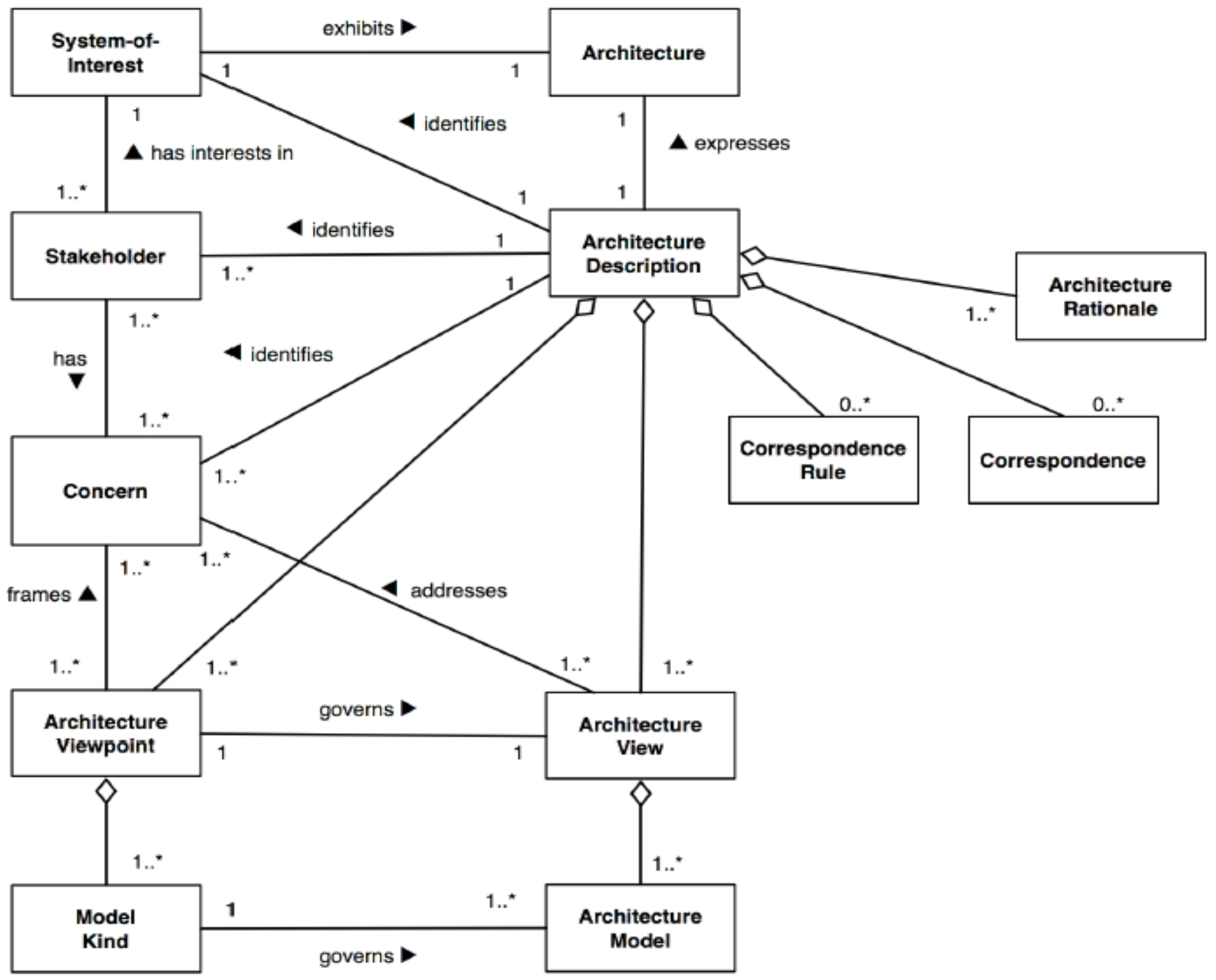
Physical View

- 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

Email: karthik.vaidhyanathan@iiit.ac.in
Web: https://karthikvaidhyanathan.com
Twitter: @karthi_ishere