BITS Pilani

Software Architecture

Introduction

Author: Viswanathan Hariharan
Edited by Vijayarajan A

Contents

- What is software architecture?
- Importance of software architecture
- Difference between Architecture and Design
- Architecture patterns
- Characteristics of good architecture
- Challenges in software architecture
- Role of an architect

- About this course
- Faculty Introductions

Objective of the course

No	Course Objective
CO1	To enable software engineers to architect software systems using industry best practices
CO2	To enable project managers to understand techniques of software architecture, and help them take appropriate decisions
CO3	To enable software professionals to take up research activities in the domain of software architecture

Learning outcomes

No	Learning Outcome
LO1	Ability to identify architecturally significant requirements and apply appropriate tactics to address them
LO2	Ability to determine appropriate architecture patterns for given requirements
LO3	Ability to document architecture that meets the needs of stakeholders
LO4	Ability to analyse architecture and determine its appropriateness given the requirement and determine risks
LO5	Awareness of best practices in design of cloud based applications, distributed applications and mobile applications
LO6	Awareness of new technologies and their architecture and understanding of situations when to use these technologies
LO7	Ability evaluate the cost and benefit of different architecture options to aid in decision making

Methodology

- Lectures
- Activities
- Assignments
- Self study

Evaluation method

- 2 Quizzes
- 2 Assignments
- Mid term exam
- End term exam

Handout

Refer to e-Learn Portal (Taxila)

Faculty Introduction

- M Tech (CS), MBA
- 43 Years of Industry Experience
  - Wipro (Chief Executive, Health Science)
  - GE Medical (Vice President)
  - HP (Vice President)
  - InnAccel (Founder & CTO) ✅
- Information Technology, MedTech
  - System SW (Compiler, DB Development)
  - Electronic Medical Record
  - MedTech (Devices and SW)

What is software architecture?

Can we try to define ‘Software architecture’?

Evolution of SW Architecture

We design and implement information systems to solve problems and process data.

As problems become larger and more complex and data becomes more voluminous, so do the associated information systems

- Structured programming, Data Structure, Higher Level languages, software engineering, Object Oriented etc

Computing become Distributed, on the cloud, Mobile as a front end

As the problem size and complexity increase, algorithms and data structures become less important than getting the right structure for the information system.

Specifying the right structure of the information system becomes a critical design problem itself

< Example from Construction Industry>

What is software architecture?

Software architecture depicts the organization of software components and how the software system works. It shows:

- The arrangement of software components
- Connection and interaction between software components
- Distribution of software components across different computing systems

<u>Formal Definition of Architecture</u>

the architecture of a system describes its gross **structure**. This structure illuminates the top-level **design decisions**, including things such as how the **system is composed of** interacting **parts,** what are the principal **pathways of interaction**, and what are the **key properties of the parts and the system** as a whole. Additionally, an architectural description includes sufficient **information to allow high-level analysis and critical appraisal**.

SW Architecture another Defn.

- The **structure** of the system refers to architecture style (or styles) the system is implemented in (such as microservices, layered, or microkernel)”

- Architecture **characteristics** refers to the “-ilities” that the system must support”

- Architecture **decisions** define the rules for how a system should be constructed. For example, only the business and services layers can access the database restricting the presentation layer from making direct database calls.

- **Design principle** is a guideline rather than a hard-and-fast rule. For example, the design principle may state that the development teams should leverage asynchronous messaging between services within a microservices architecture to increase performance.

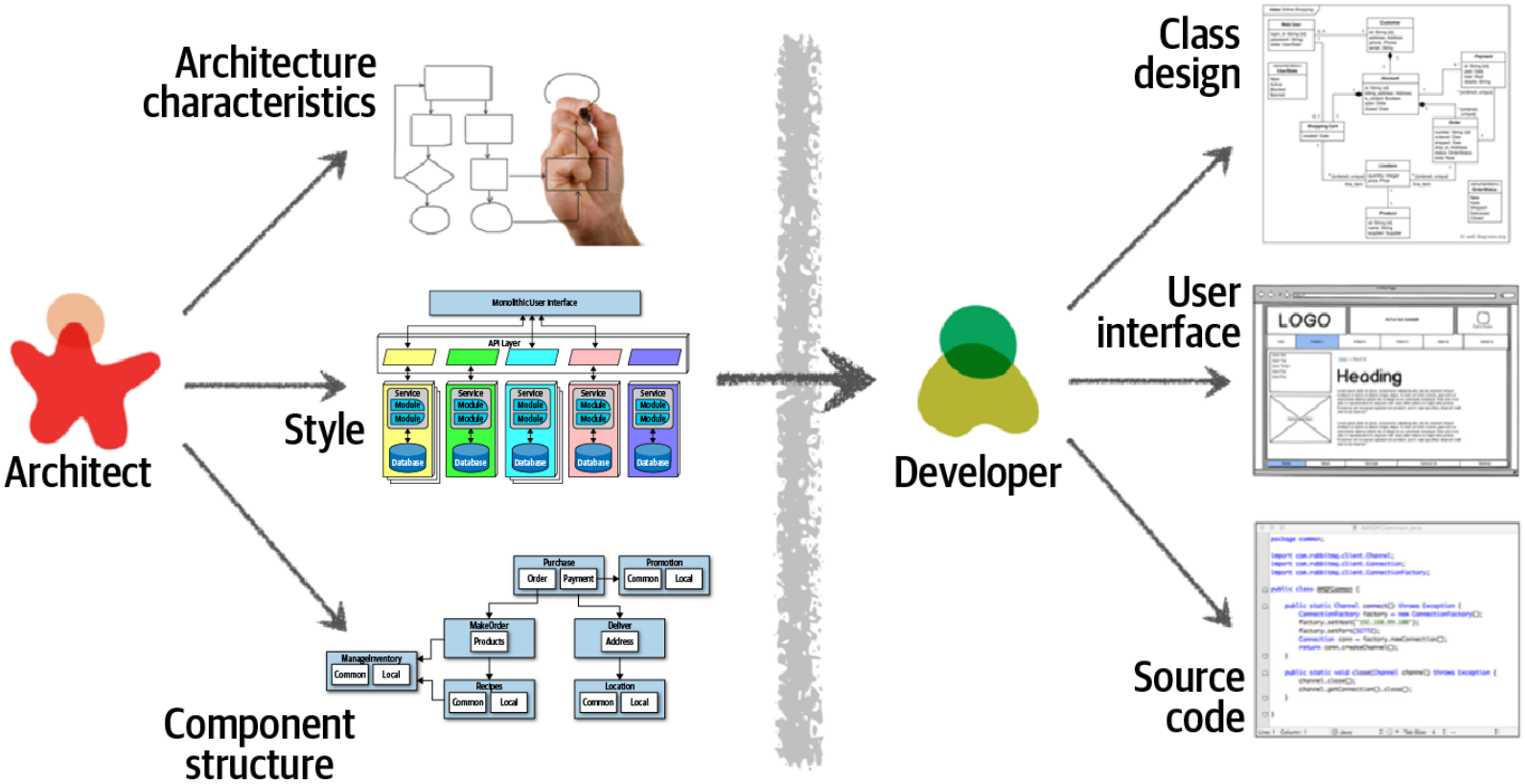
Role of Architect

- Make architecture decisions
- Continually analyze the architecture
- Keep current with latest trends
- Ensure compliance with decisions
- Diverse exposure and experience
- Have business domain knowledge
- Possess interpersonal skills
- Understand and navigate politics

Difference between architecture and design

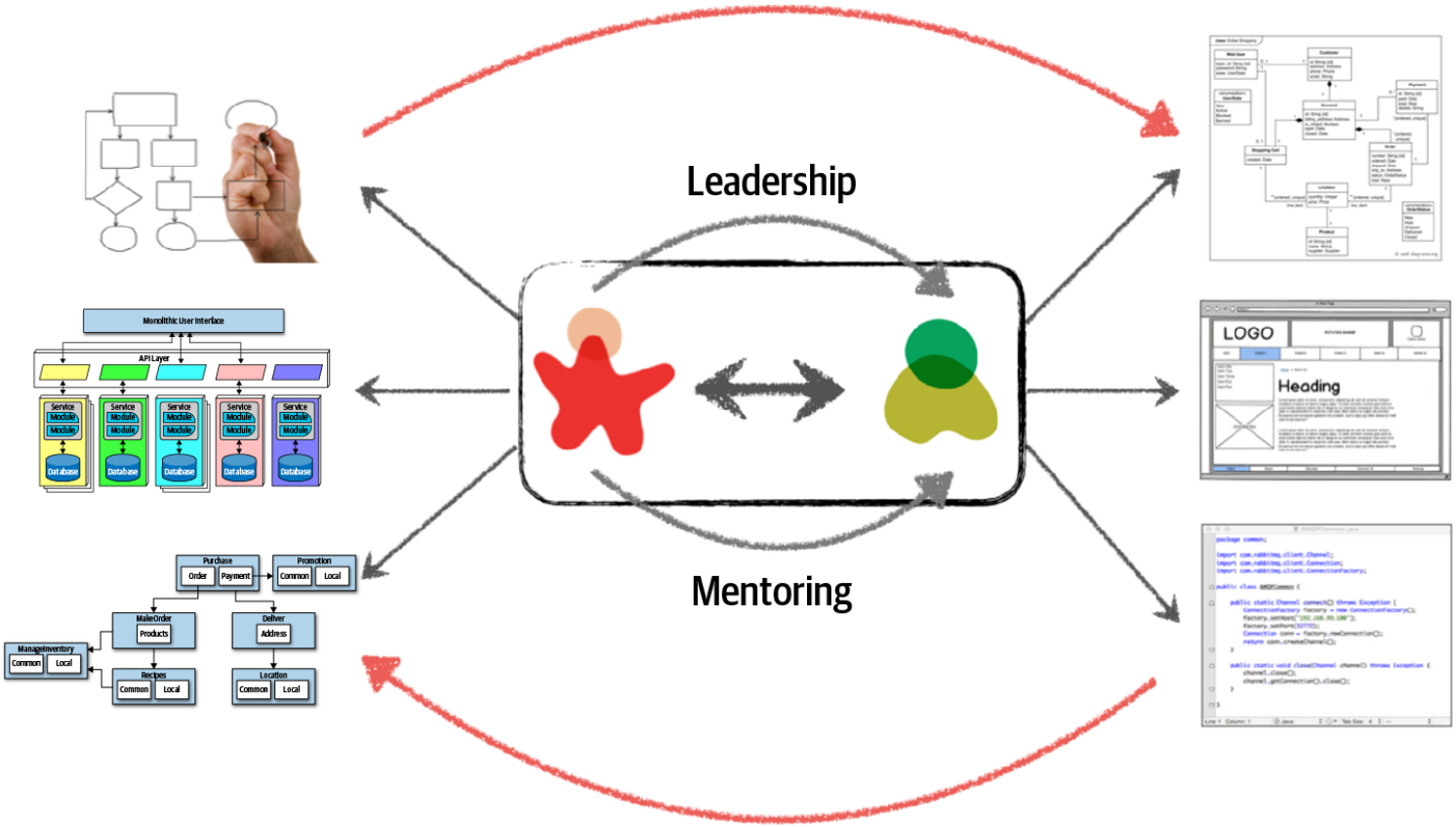
Architecture	Design
• Deals with design at a system	• Deals with design at Module level
• Based on business goals, requirements and constraints	• Based on purpose of module
• Involves decomposing the system into components and their interactions	• Involves designing objects within a module and interactions between modules

Difference between architecture and design



Architecting Used To be a Phase

Difference between architecture and design



Architecting is integrated in the Development Process

Difference between architecture and design

"a developer, who must have a significant amount of technical depth to perform their job, a software architect must have a significant amount of technical breadth to think like an architect and see things with an architecture point of view."

Architecture - Trade Offs

- There are no right or wrong answers in architecture—only trade-offs
- Everything in software architecture has a trade-off: an advantage and disadvantage.
- Thinking like an architect is analyzing these trade-offs, then asking "which is more important: extensibility or security?"
- The decision between different solutions will always depend on the business drivers, environment, and a host of other factors."

Architecting has become iterative

Unknown unknowns are the nemesis of software systems.

Many projects start with a list of known unknowns.

However, projects also fall victim to unknown unknowns: things no one knew were going to crop up yet have appeared unexpectedly.

This is why all “Big Design Up Front” software efforts suffer: architects cannot design for unknown unknowns.

All architectures become <u>iterative</u> because of unknown unknowns, Agile just recognizes this and does it sooner”

Software architecture

Now let us look at an example of software architecture…

Software architecture consists of many diagrams

- Context diagram
- Logical diagram / Component & Connection diagram
- Physical diagram / Deployment diagram
- A diagram to explain a scenario

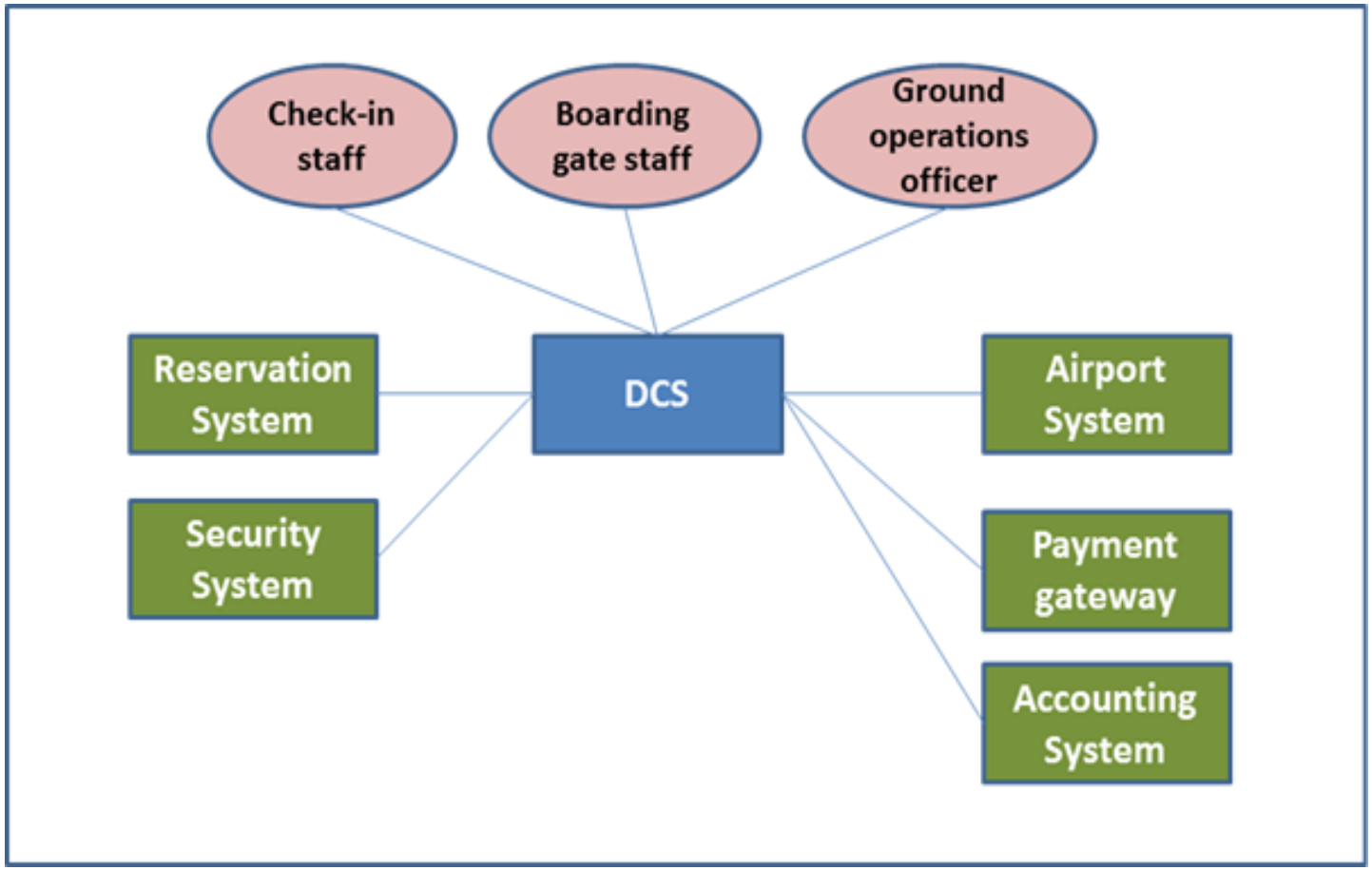
- Each diagram provides a different perspective

- **Software architecture = These diagrams + associated descriptions**

Context diagram

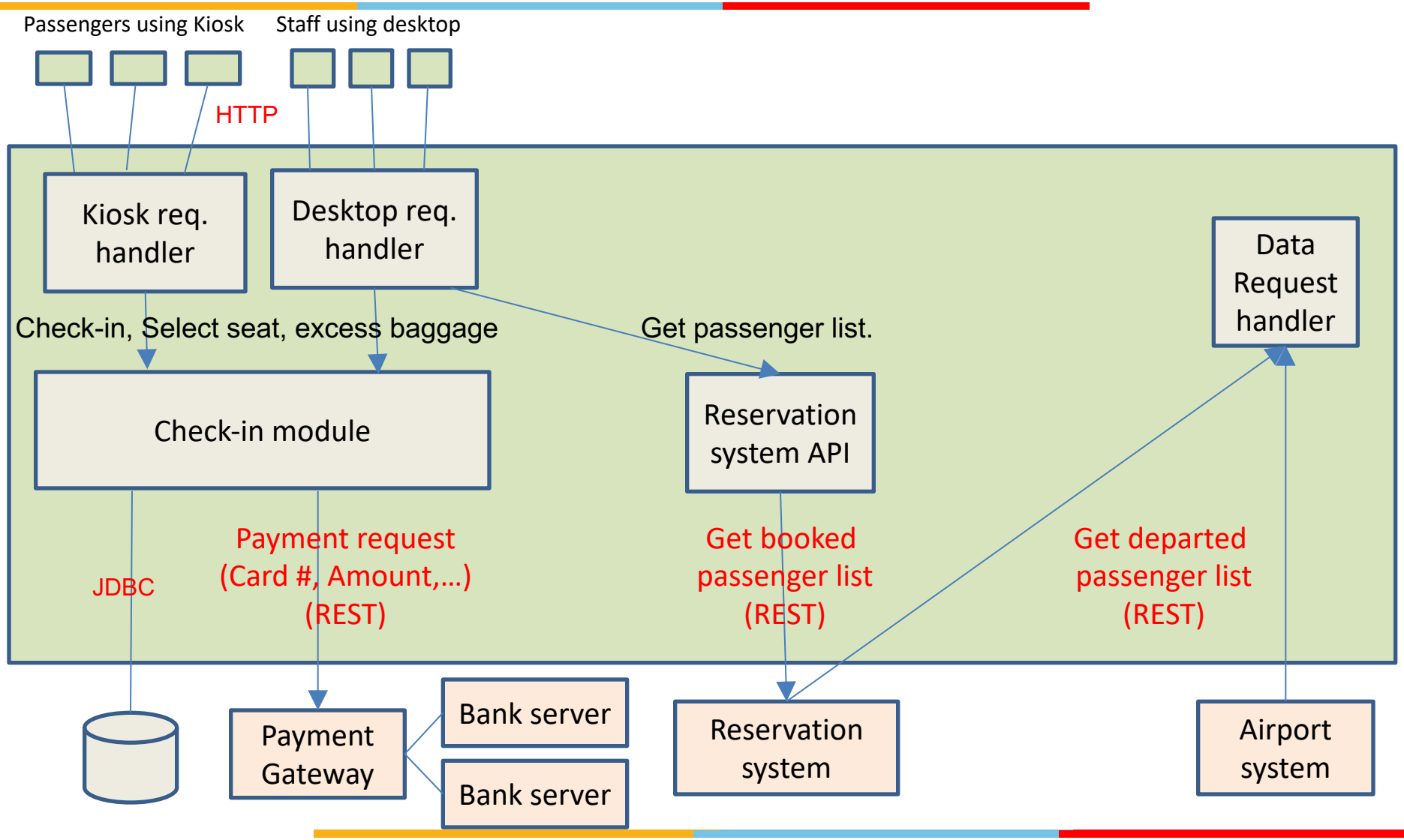
Shows how the software fits in the overall system

Departure Control System at Airport



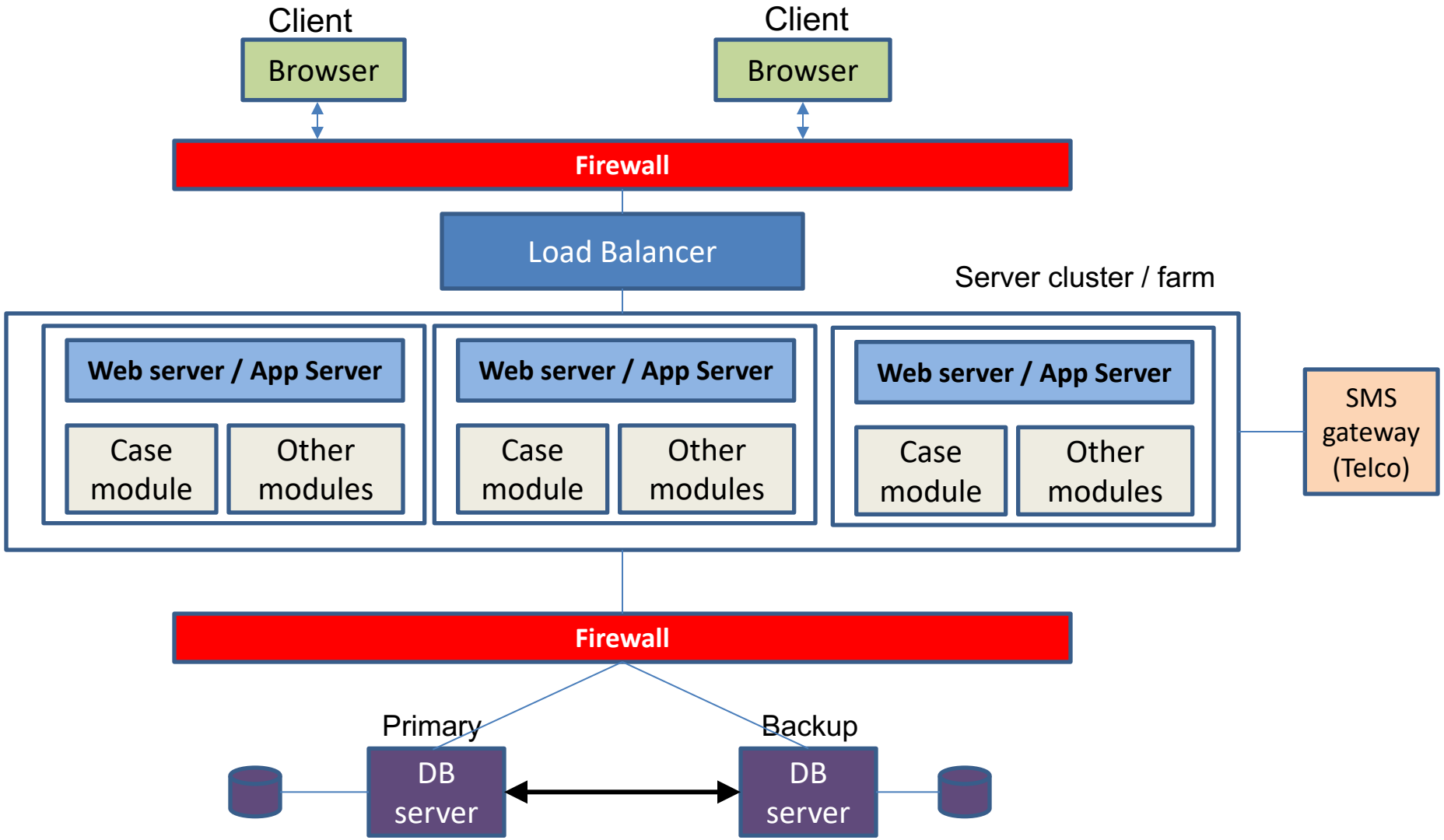
Logical diagram / Component & Connection view

Shows different components and their logical connection with other components



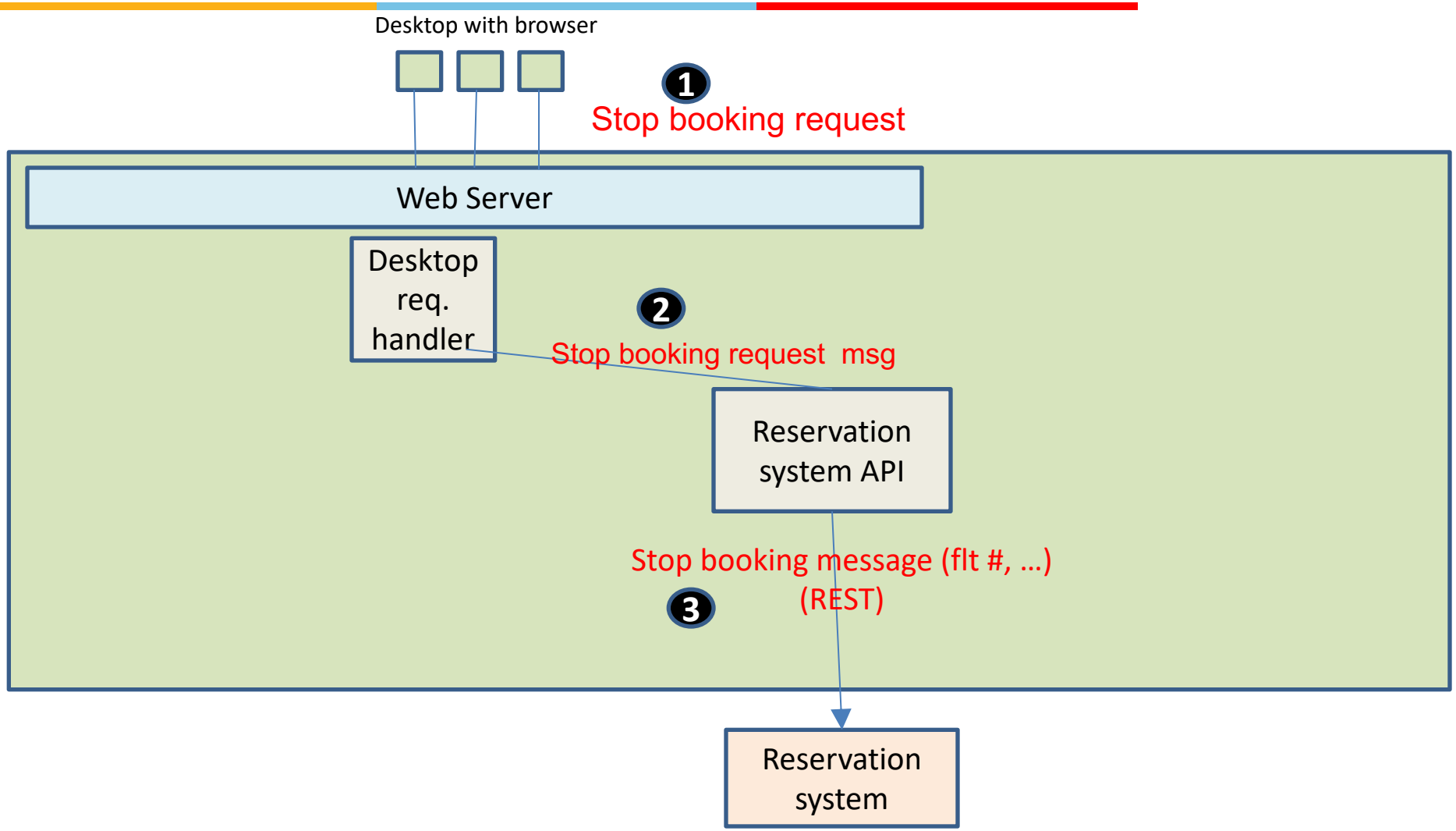
Physical diagram / Deployment diagram

Shows distribution of sw components across computing units



A diagram (view) to explain a scenario:

Ex. Show how the 'Stop booking' request works



Why is software architecture important?

- Lays foundation for future work - detailed design, development & testing

- Helps evaluate if approach is correct
  - Is the system secure against hacking, snooping, etc. (firewall, encryption)?
  - Is the system easy to maintain (Modular, Layered, Reusable components, etc.)?
  - Will the system provide desired response time (sufficient servers, replicated data, caching)?

- Helps stakeholders understand how the system will work. Stakeholders are sponsors, developers, operations staff, etc.

Architecture structures

The 3 structures correspond to 3 broad types of decisions that an architectural design involves:

- **Module structure**: How is the system to be structured as a set of code units?
- **Component & Connection structures**: How is the system to be structured as a set of runtime components and interactions between them?
- **Deployment structure**: How is the system to relate to non-software elements in its environment? (CPU, file systems, networks, development teams, etc)

Module Structure

Module structures allow us to answer questions such as these:

- What is the primary functional responsibility assigned to each module?
- What other software elements is a module allowed to use?
- What other software does it actually use and depend on?
- What modules are related to other modules by generalization or specialization (i.e., inheritance) relationships?

Component-and-connector Structures

Component-and-connector views help us answer questions such as these:

- What are the major executing components and how do they interact at runtime?
- What are the major shared data stores?
- Which parts of the system are replicated?
- How does data progress through the system?
- What parts of the system can run in parallel?
- Can the system's structure change as it executes and, if so, how?

Component-and-connector views are crucially important for asking questions about the system's runtime properties such as performance, security, availability, and more.

Allocation structures

Allocation views help us answer questions such as these:

- What processor does each software element execute on?
- In what directories or files is each element stored during development, testing, and system building?
- What is the assignment of each software element to development teams?

Architectural Patterns

Architectural elements can be composed in ways that solve particular problems.

- The compositions have been found useful over time, and over many different domains
- They have been documented and disseminated.
- These compositions of architectural elements, called architectural patterns.
- Patterns provide packaged strategies for solving some of the problems facing a system.

A common module type pattern is the Layered pattern.

- When the uses relation among software elements is strictly unidirectional, a system of layers emerges.
- A layer is a coherent set of related functionality.
- Many variations of this pattern, lessening the structural restriction, occur in practice.

Architectural Patterns

Common component-and-connector type patterns:

Shared-data (or repository) pattern.

- This pattern comprises components and connectors that create, store, and access persistent data.
- The repository usually takes the form of a (commercial) database.
- The connectors are protocols for managing the data, such as SQL.

Client-server pattern.

- The components are the clients and the servers.
- The connectors are protocols and messages they share among each other to carry out the system's work.

Architectural Patterns

Common allocation patterns:

Multi-tier pattern

- Describes how to distribute and allocate the components of a system in distinct subsets of hardware and software, connected by some communication medium.
- This pattern specializes the generic deployment (software-to-hardware allocation) structure.

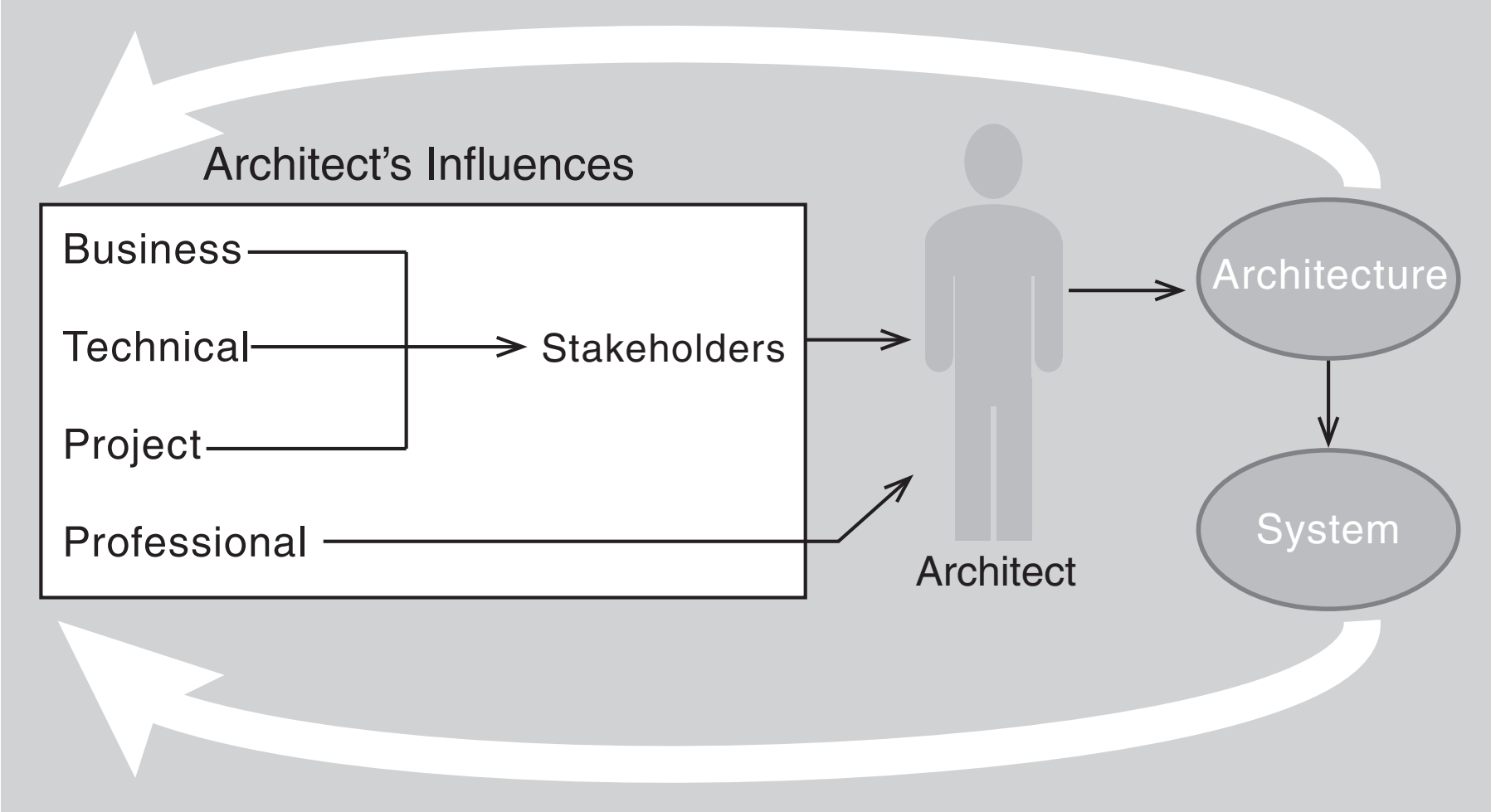
Competence center pattern and platform pattern

- These patterns specialize a software system's work assignment structure.
- In competence center, work is allocated to sites depending on the technical or domain expertise located at a site.
- In platform, one site is tasked with developing reusable core assets of a software product line, and other sites develop applications that use the core assets.

Characteristics of good software architecture

- Meets the functional & non-functional requirements
- Reduces complexity and easy to understand
- Easy to extend and modify
- Not over engineered
- Cost effective

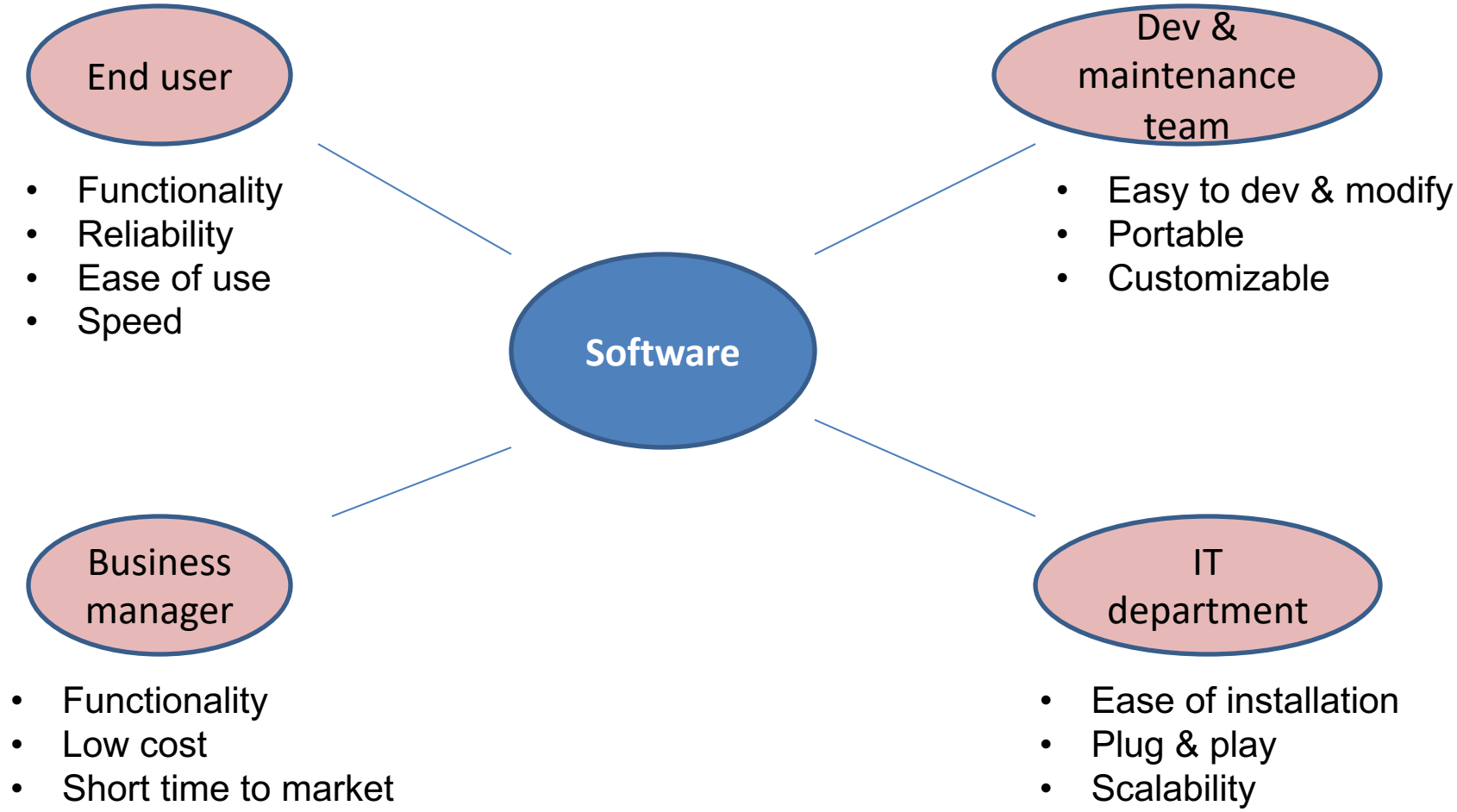
Architecture Influence Cycle



Input for software architecture

- What are the inputs needed for doing software architecture?
- Business goals. Provide a ‘Unified view’ of the customer to the Bank staff, by presenting information from disparate systems such as Savings bank system, Loan system, Credit card system, etc. This will enable be better customer service by staff.
- Functional requirements. System should support customer service functions, loyalty program functions, customer needs prediction function, etc.
- Quality attributes (Stakeholder expectations). Need to have <3 sec response time with a peak load of 1,000 concurrent users and should have an up-time of 99.95%
- Constraints: Need to work with legacy mainframe systems, Data should reside within Europe

Expectations of different stakeholders



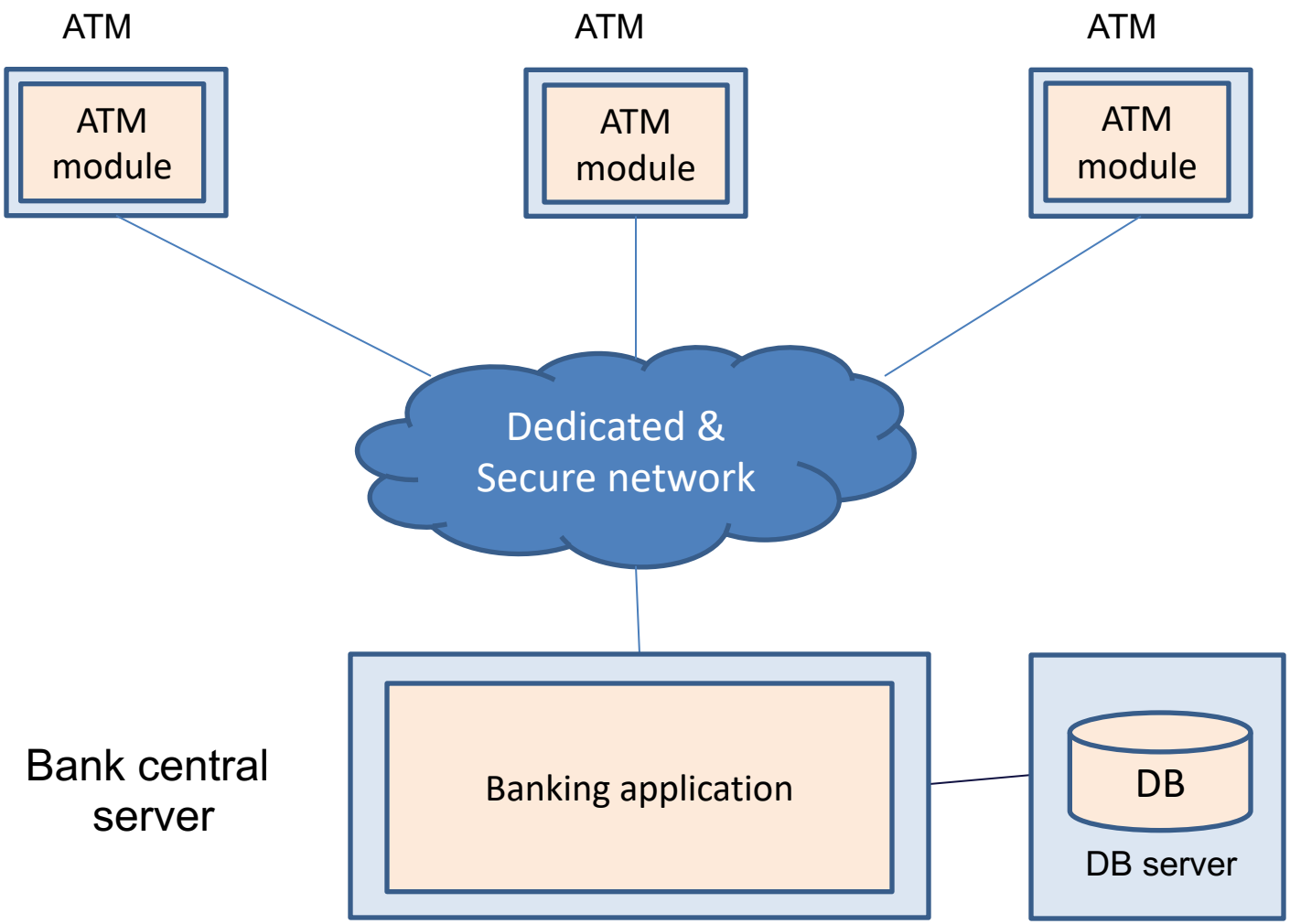
Exercise: Draw architecture of a Retail banking system with ATMs

1. Identify the different physical components
2. Determine the software components that reside in them
3. Determine the communication between the components

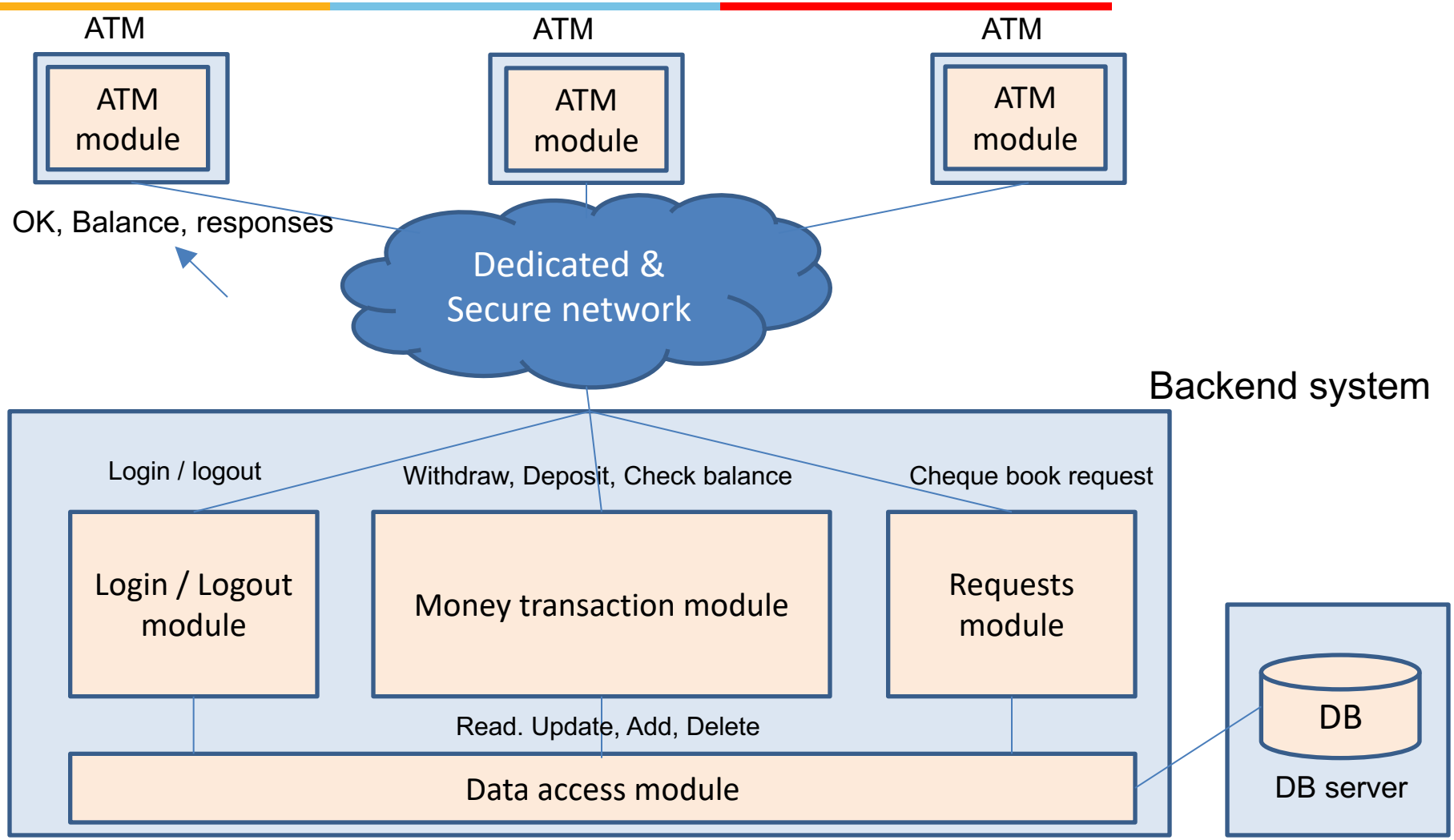
Architecture of a banking software supporting ATM

See next slide

High level diagram



Component and Connection diagram



Message communications

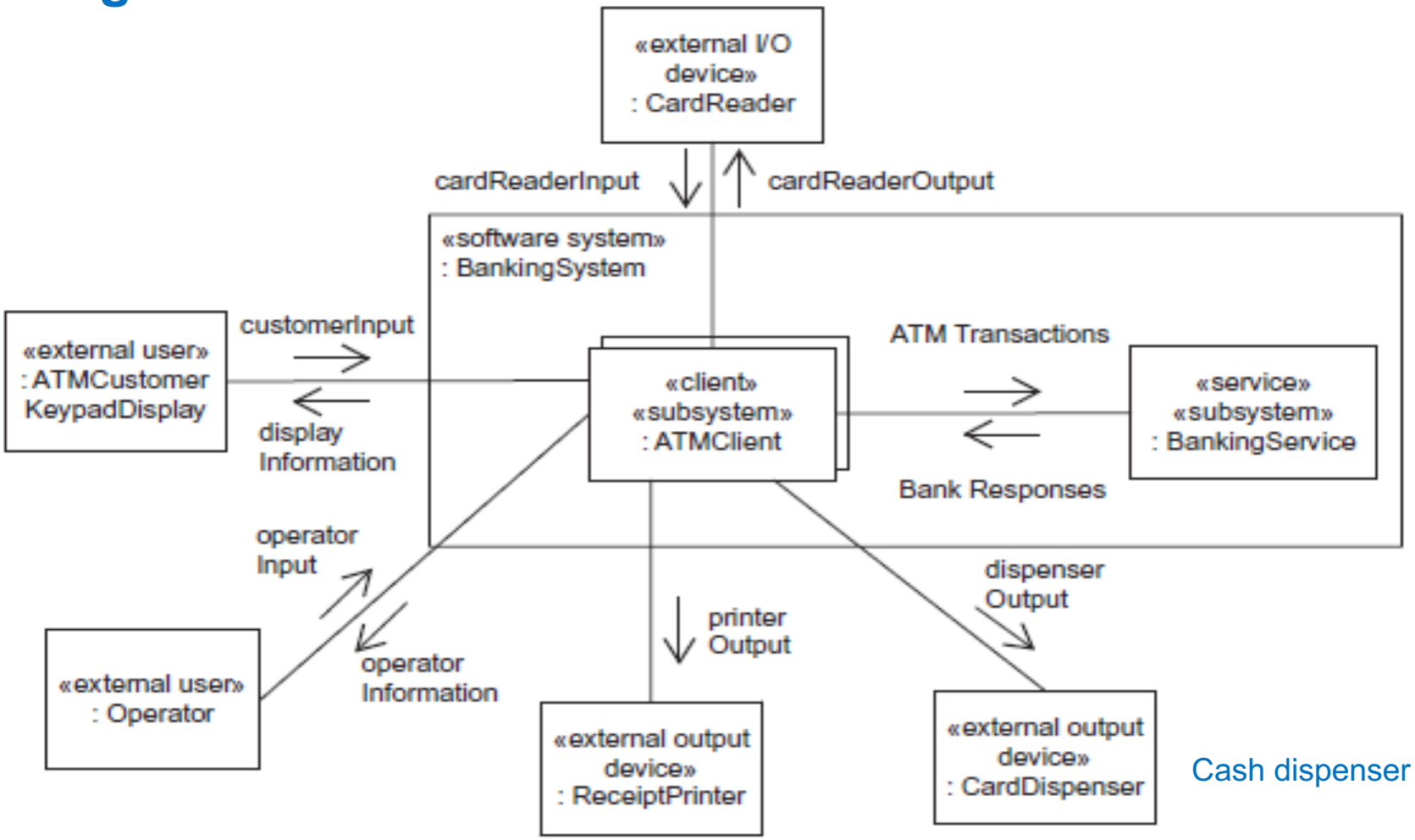


Figure 21.27. Subsystem design: high-level communication diagram for Banking System

From book: Software Modelling & design by Hassan Gomaa

Exercise

What kind of a system are you developing?

- Business system (information system)
- Real time system (control system, avionics, etc.)
- Data analytics system
- Mobile application
- Internet of Things system
- Video streaming system
- Image processing system (Medical imaging, Adobe, etc.)
- Networking (routers, SDN, etc.)
- Robotic system
- Others

Exercise

- Create a high level software architecture diagram of the system you are developing, showing its major components and their inter-relationship, and

- Upload the same in this Google folder

  https://drive.google.com/drive/folders/1_iwHUhadscnBEZwopmUWVopkahhy506e?usp=sharing

- File naming convention: <Type of system> <Name of system>.ppt

- Type of system can be one of the following:
  - Business system (information system)
  - Real time system (control system, avionics, etc.)
  - Data analytics system
  - Mobile application
  - Internet of Things system
  - Video streaming system
  - Image processing system (Medical imaging, Adobe, etc.)
  - Networking (routers, SDN, etc.)
  - Robotic system
  - Others

Appendix

Many contexts of architecture

Context	Description
Technical	• Enables achieving the quality attributes such as performance, availability, security, etc. • Depends on the technology available such as mainframe, client server, web based, Object oriented, cloud based, etc.
Project Life cycle	• Architecture is created on requirements (ASR) • It is used to develop software • It is used during testing, eg. Integration testing, performance testing, etc.
Business	• Business goals result in quality attributes such as response time of 3 seconds, uptime of 99.999% • Quality attributes influence architecture
Professional	• An architect should not only have good technical knowledge but also be able to explain stakeholders why certain quality attributes have been given higher priority (trade offs), why certain expectations are not being fulfilled

Different types of application and their architecture

- Internet based apps – Banking, eCommerce
- Mobile apps
- Real time systems – Industrial control, avionics
- Operating system
- ERP
- Networking systems
- Workflow based apps
- Content based apps

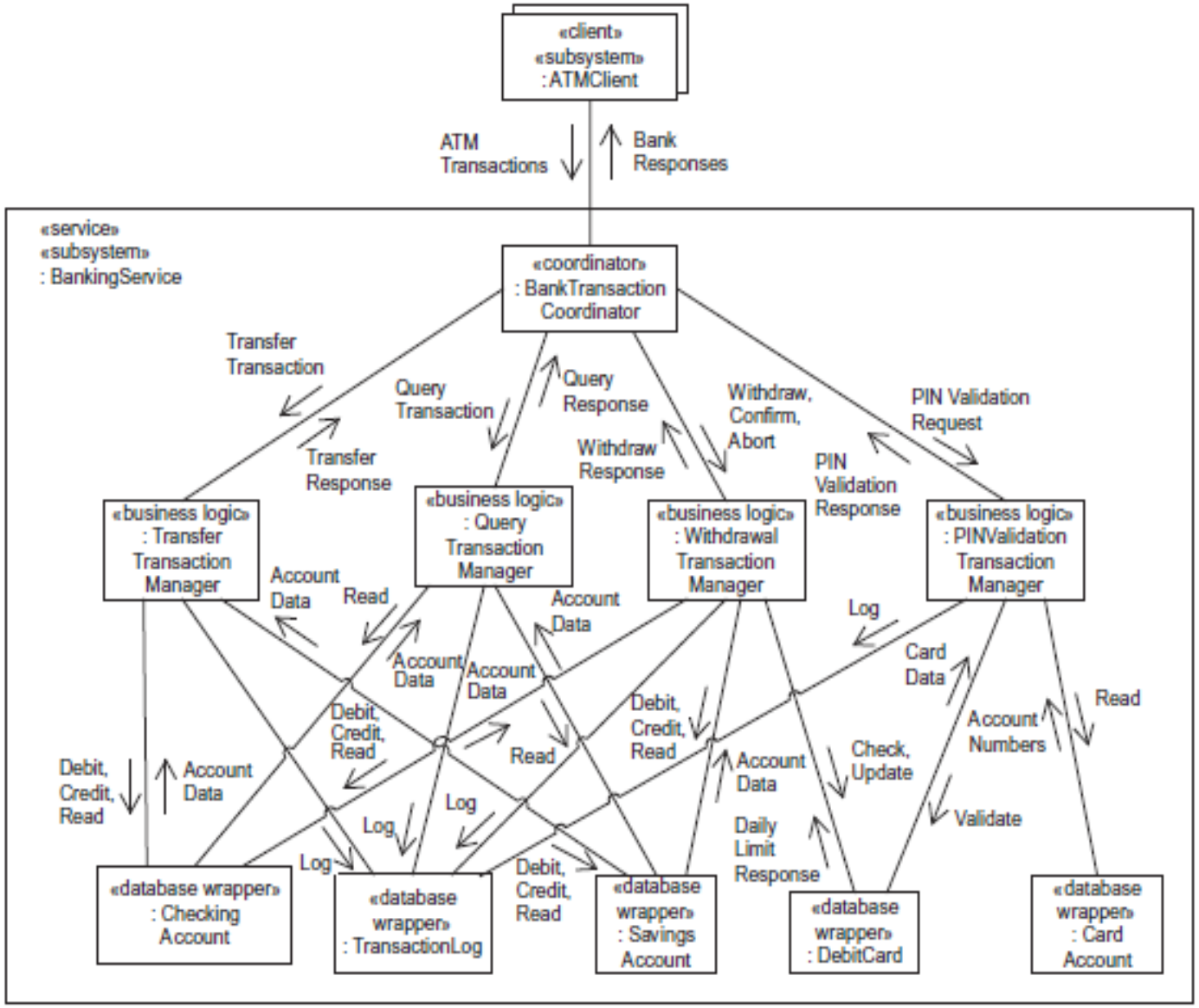


Figure 21.26. Integrated communication diagram for Banking Service subsystem

From book: Software Modelling & design by Hassan Gomaa

Real time systems – Automated guided vehicle

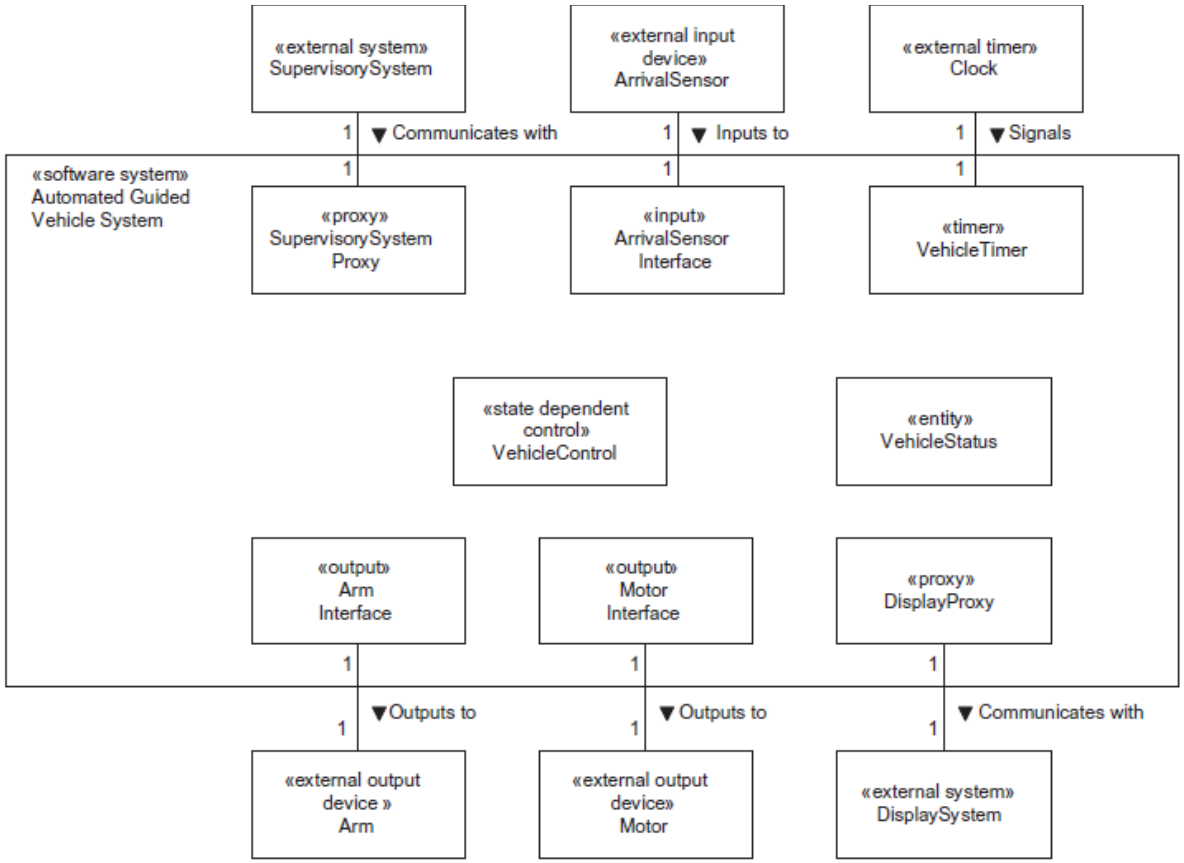


Figure 24.4. Object structuring for the Automated Guided Vehicle System

Real time system

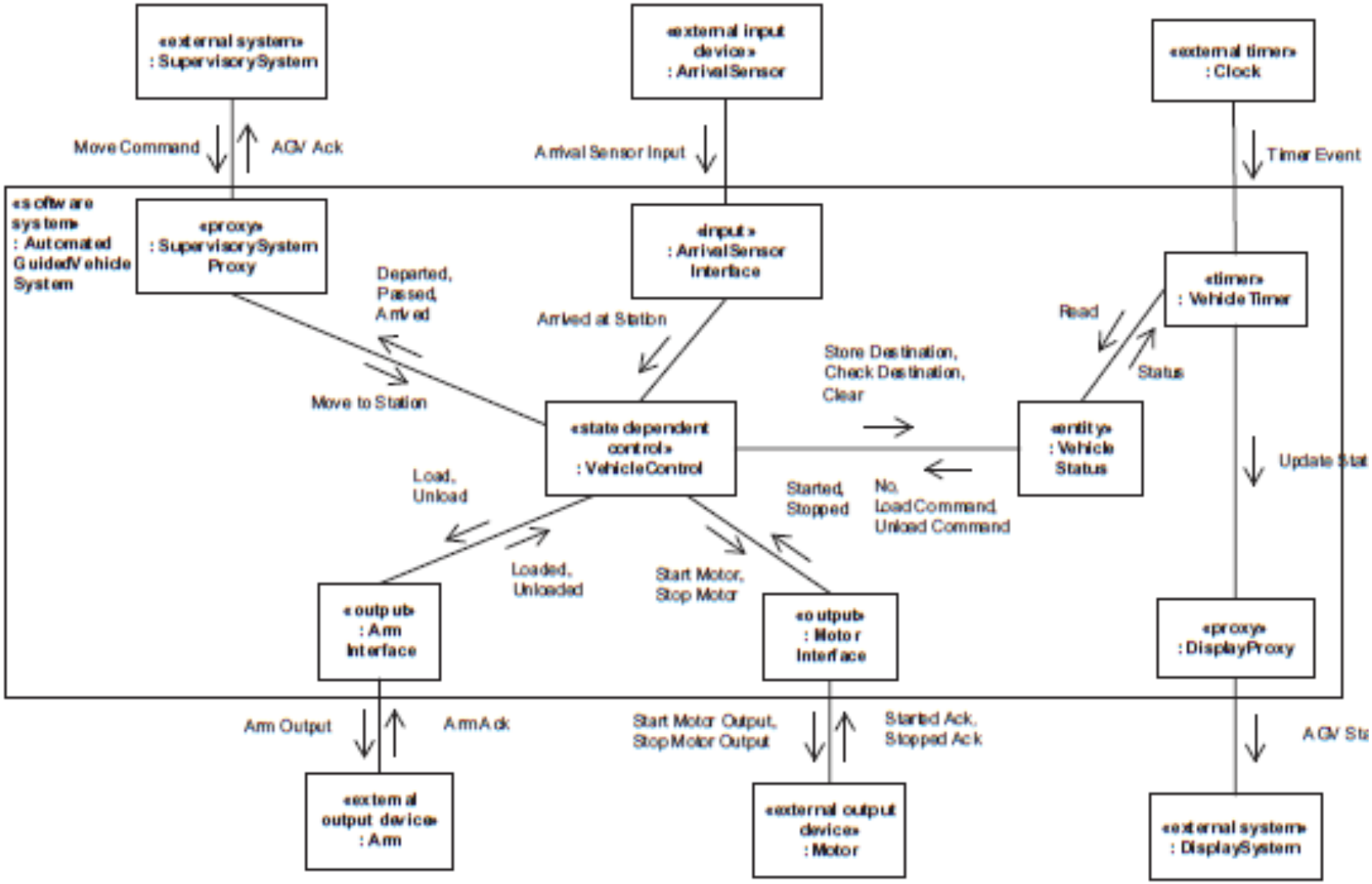


Figure 24.9. Integrated communication diagram for Automated Guided Vehicle System

Recommendations for making a good architecture

Process recommendations:

- Use a single architect or a small group with a leader, to architect
- Create a prioritized list of quality attributes
- Document using views to address concerns of most important stakeholders
- Develop in increments & get early feedback

Structural recommendations

- Create well defined modules with information hiding
- Use small number of ways to interact eg. RPC or REST or pipes
- If performance is a major concern, define performance expectation of each component in the chain

Views

- Views are meant for stakeholders to understand how the architecture addresses his / her concern
- It consists of a sub set of the software components and their relationships
- It can show
  - Structure: Ex. Class diagram, package diagram, context diagram
  - Behaviour: Ex Sequence diagram, state diagram,
- It also contains
  - Software element description
  - Element interfaces
  - Rationale

Structures

When we discuss about architecture we refer to different types of structures

- Module structure
  - Decomposition structure (System, sub-system, elements)
  - Class diagram (Classes, associations, interfaces, dependencies, inheritance)
  - Data model (ER diagram)
- Component & Connector structure
  - Interaction between components (sequence diagram, collaboration diagram, interaction mechanisms such as Call return, message queues, REST)
  - Process synchronization (semaphores, critical section)
  - Shared data stores (Database and access mechanisms)
- Allocation structure
  - Processors and software elements in them
  - Directories and files and what software elements they contain
  - Assignment of software elements to software teams

The 3 structures correspond to 3 broad types of decisions that an architectural design involves:

- How is the system to be structured as a set of code units?
- How is the system to be structured as a set of runtime components and interactions between them?
- How is the system to relate to non-software elements in its environment? (CPU, file systems, networks, development teams, etc)

Different types of software

Type of application	Example
Enterprise apps	Banking, Telecom, Airline, Retail
Industrial control, avionics	Monitoring & controlling a power plant or chemical factory Monitoring and controlling an aircraft
Operating system	Unix, Android, iOS
Networking systems	TCP/IP, VPN, Firewalls, Routers
Workflow based apps	Loan processing, Insurance claim processing, Invoice processing
Portals & Content management systems	Newspaper website

Different types of software & expectations from stakeholders

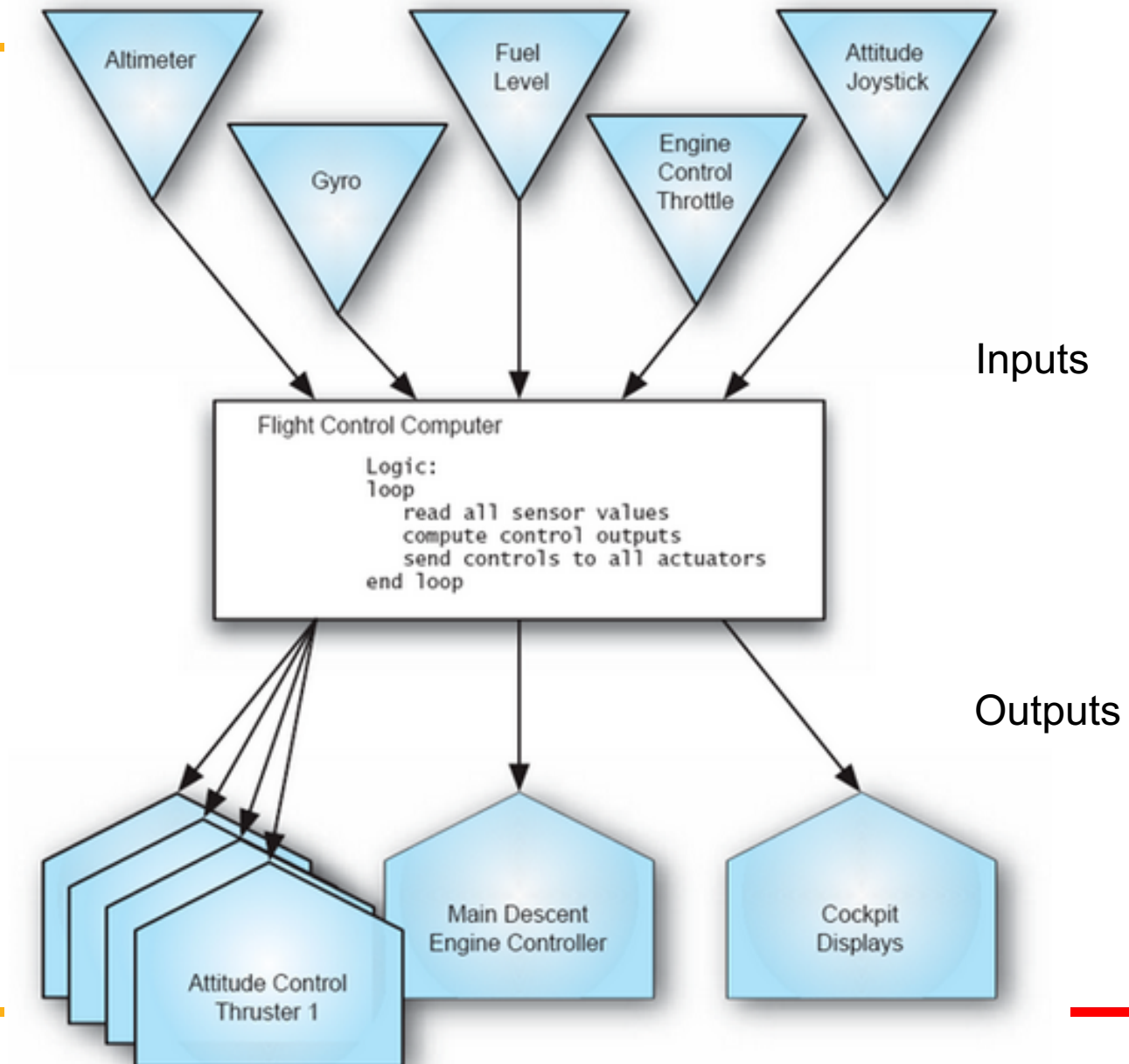
Type of application	Expectations
Enterprise apps – Banking, Telecom, Airline, Retail	Reliability, security, performance
Industrial control, avionics	Time critical, very reliable, life threatening if it fails
Operating system	Reliable, support different devices, high performance, muti-processing, security
Networking systems	Fault tolerant, secure, performance
Workflow based apps	Configurable, business rules
Portals & Content management systems	Personalization, security, data management

Homework activity

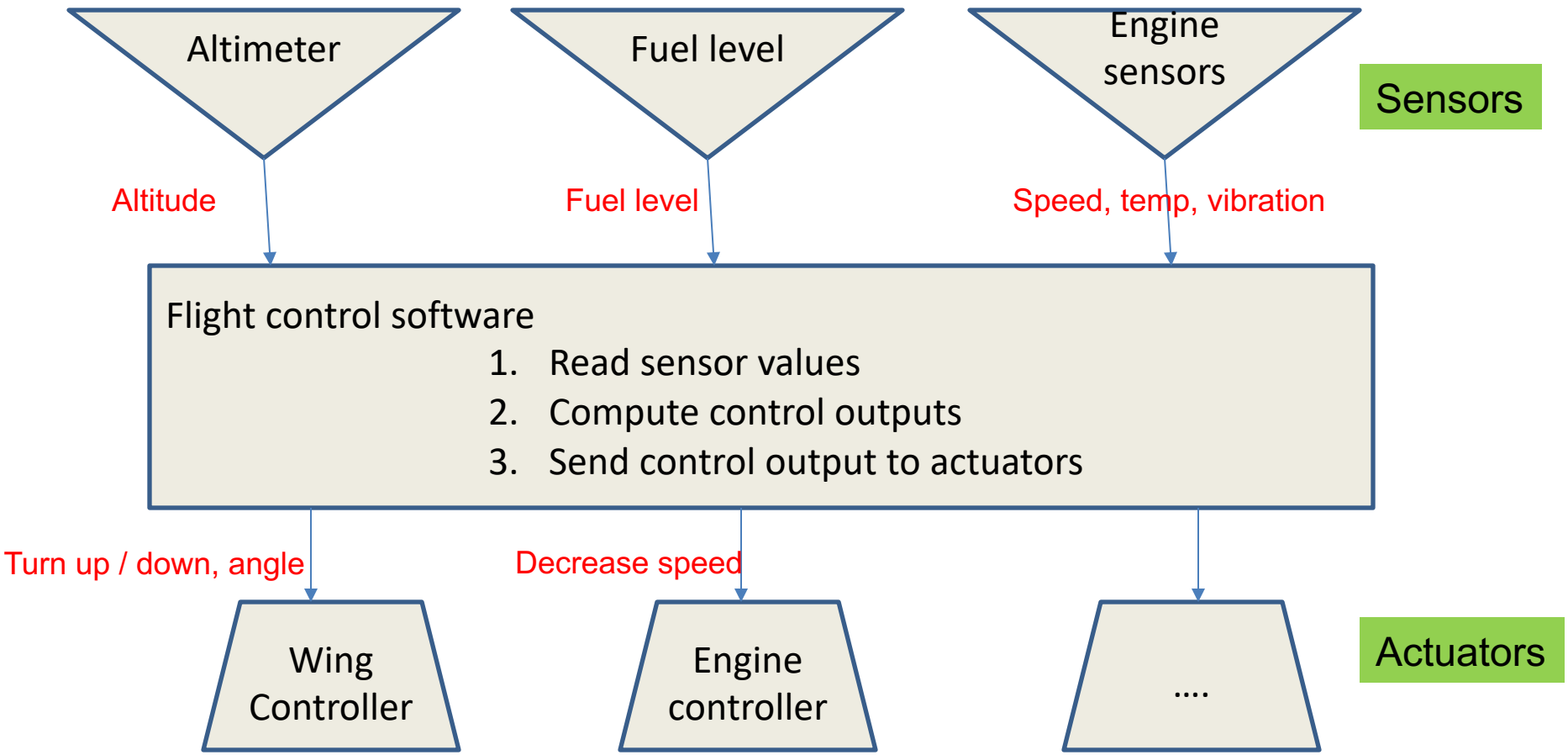
1. Get a sample architecture diagram of a software in your company

Real time system architecture

Flight control system

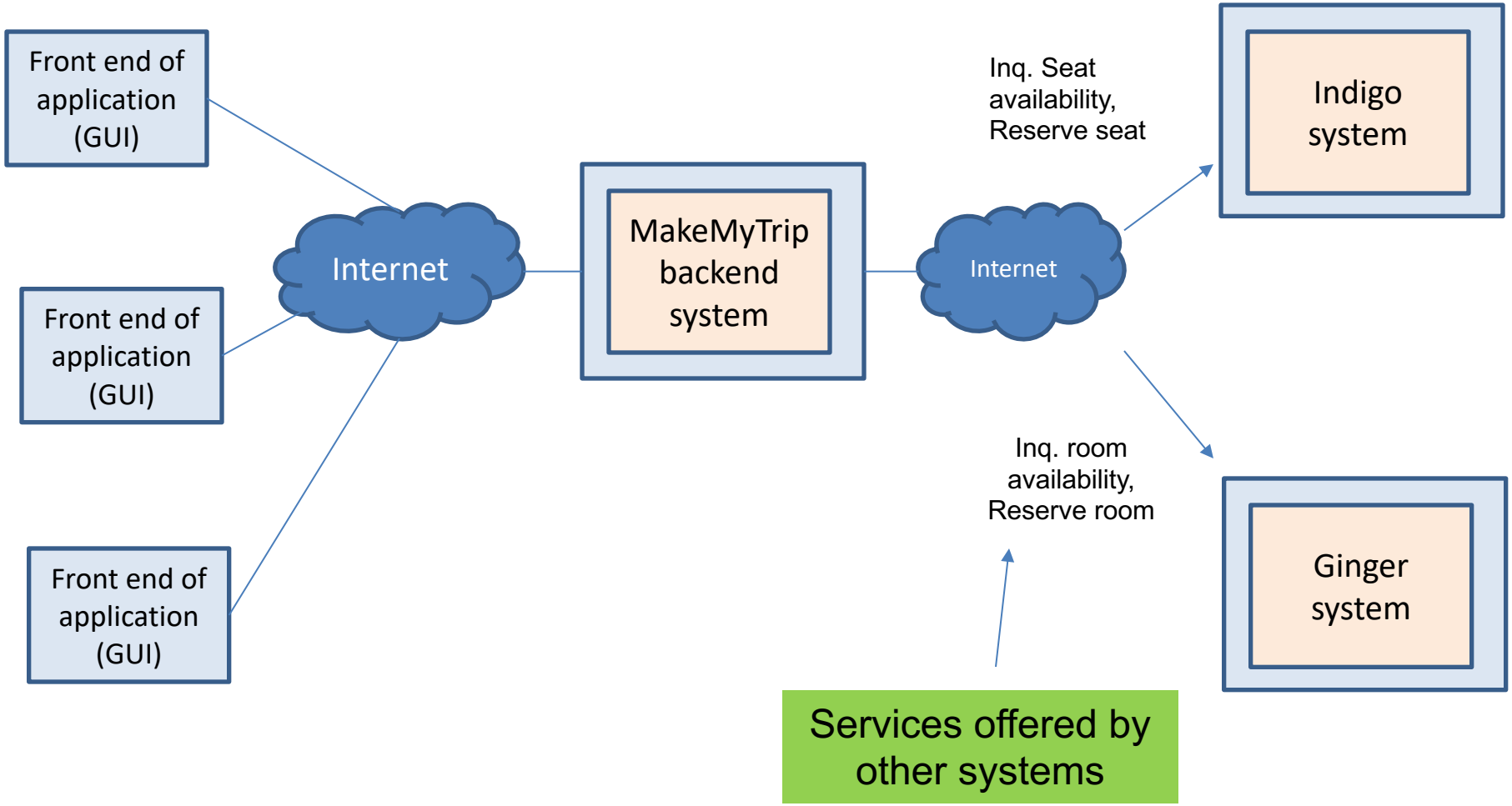


Real time software architecture



Service Oriented Architecture

Example: MakeMyTrip.com



Layered pattern:
Example: Architecture of an Information system

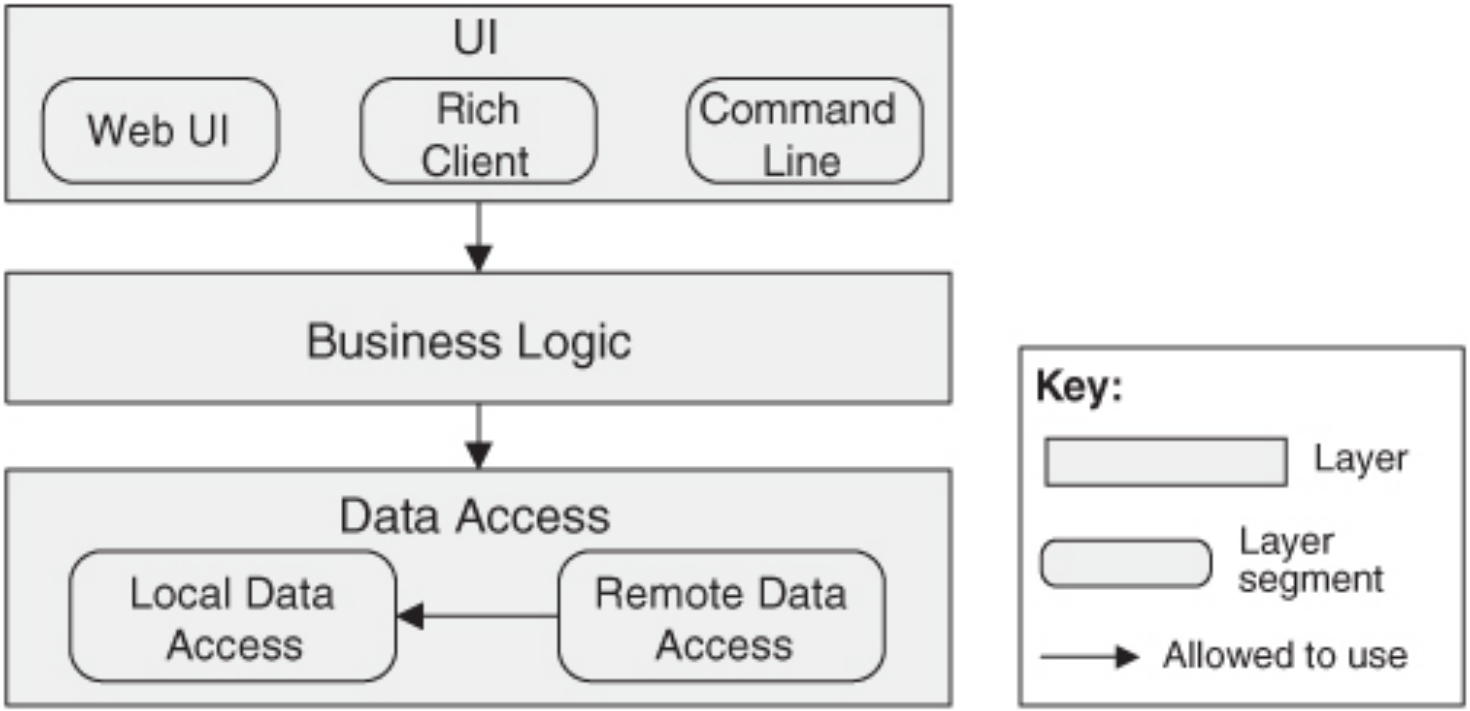


Figure 13.5. Layered design with segmented layers

Ref: Software Architecture in Practice by Len Bass and others