BITS Pilani

Software Architecture

Introduction

Viswanathan Hariharan

Contents

What is software architecture?

Before we define this, let us look at examples of architecture from construction industry

Example of a building architecture

Exclusively on Housing

Example of a building architecture

Based on these pictures, how can we define building architecture?

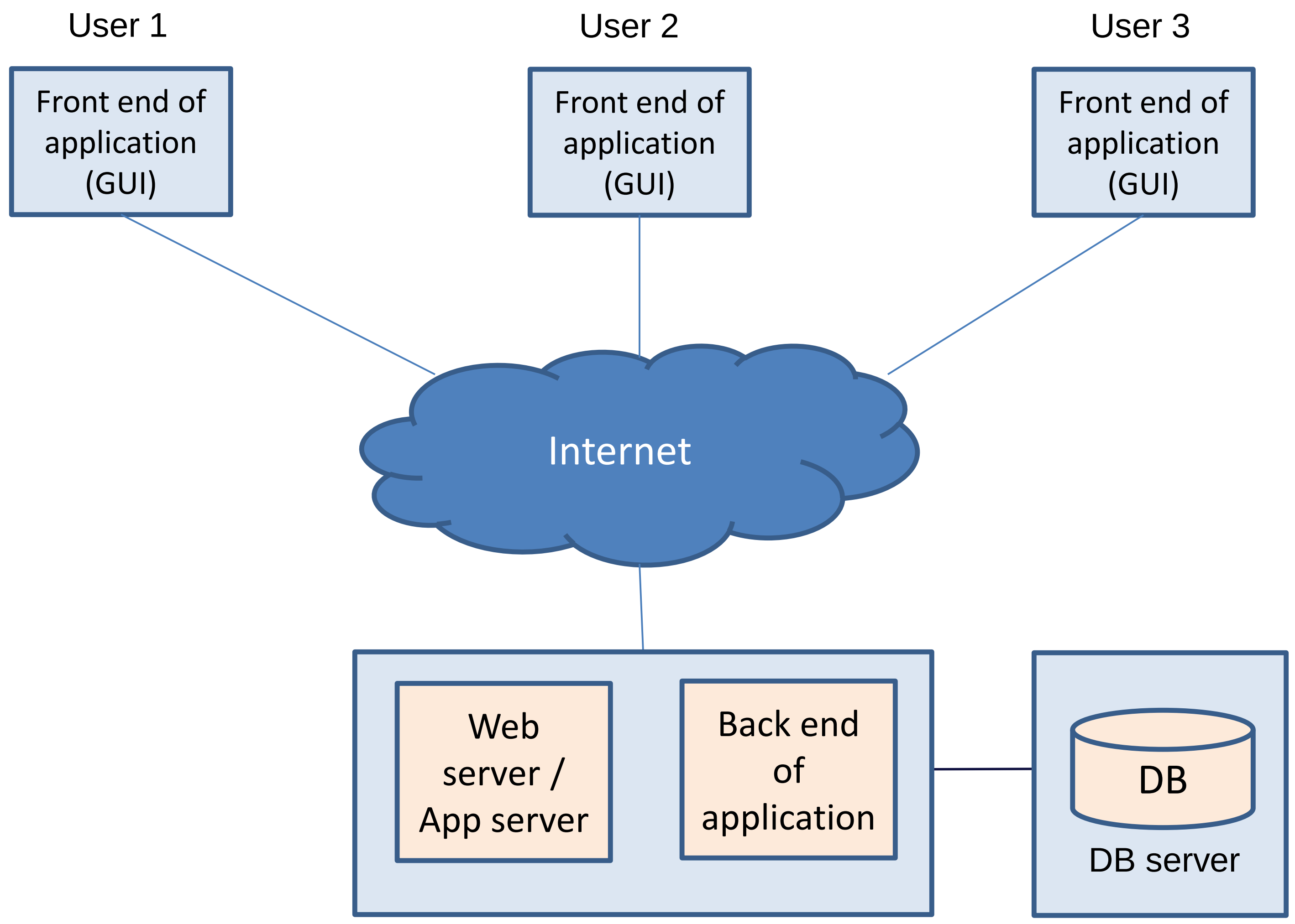
So, what is architecture of a building?

- Blue print of the building, that shows
  - Layout of the building
  - Different sections of the building
  - Relationships between different sections
- Can be High level or detailed

Software architecture

Now let us look at examples of software architecture…

Web appl architecture


User 1
User 2
User 3
Front end of application (GUI)
Front end of application (GUI)
Front end of application (GUI)
Internet
Web server / App server
Back end of application
DB
DB server

What is software architecture?

Can we try to define ‘Software architecture’?

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

Software architecture consists of many diagrams

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

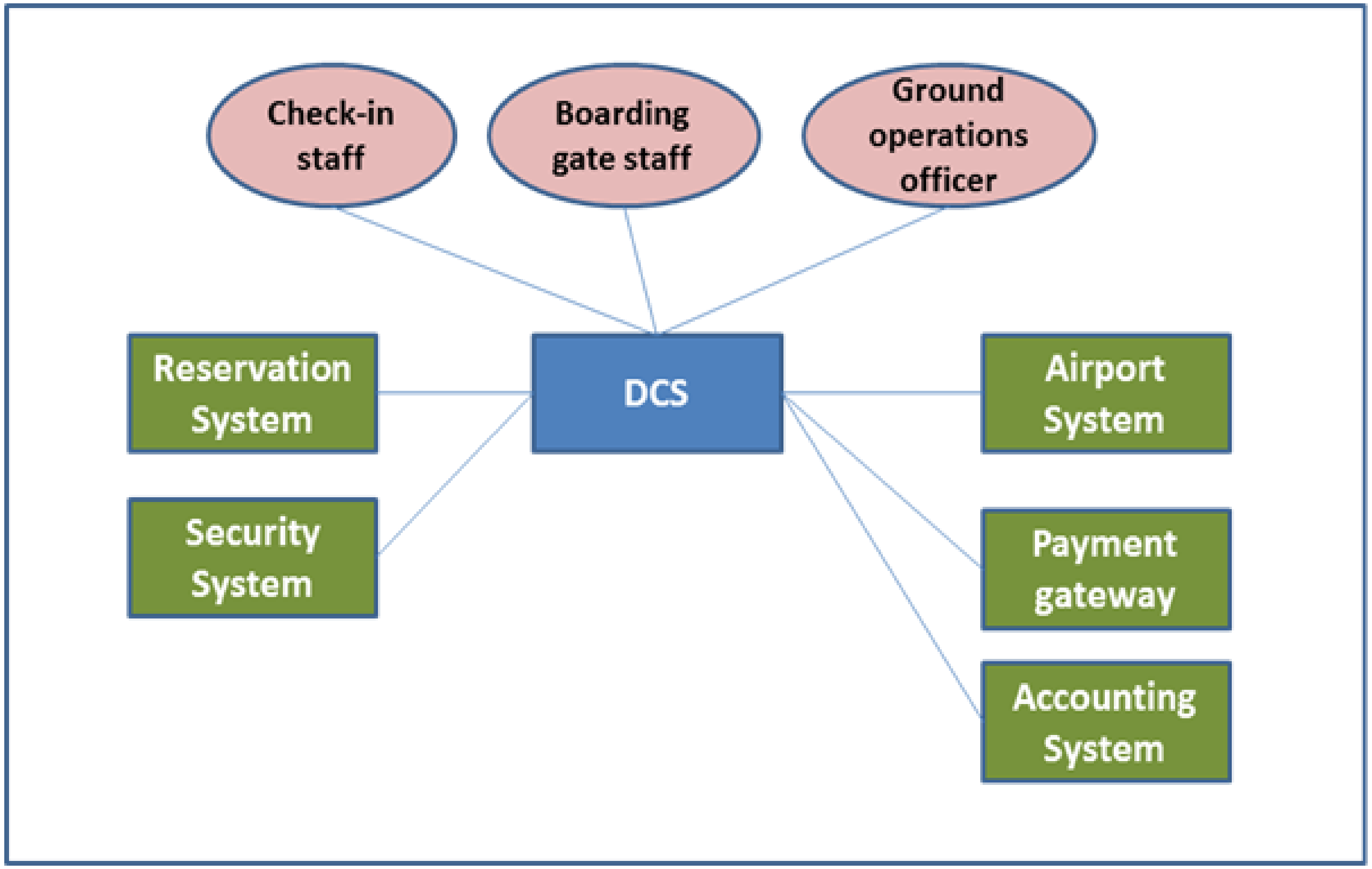
- Each diagram provides a different perspective

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

Context diagram

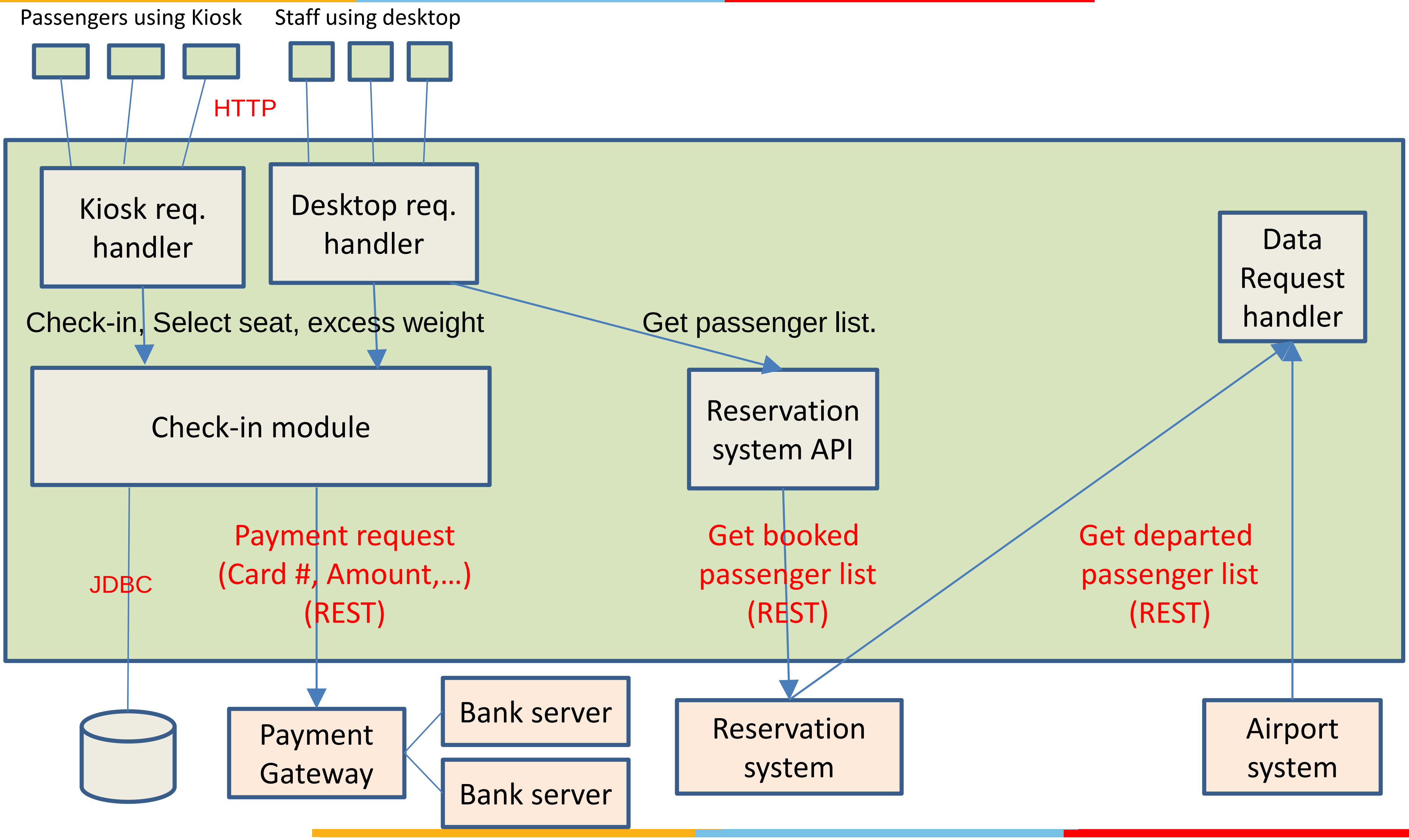
Shows how the software fits in the overall system

Departure Control System at Airport



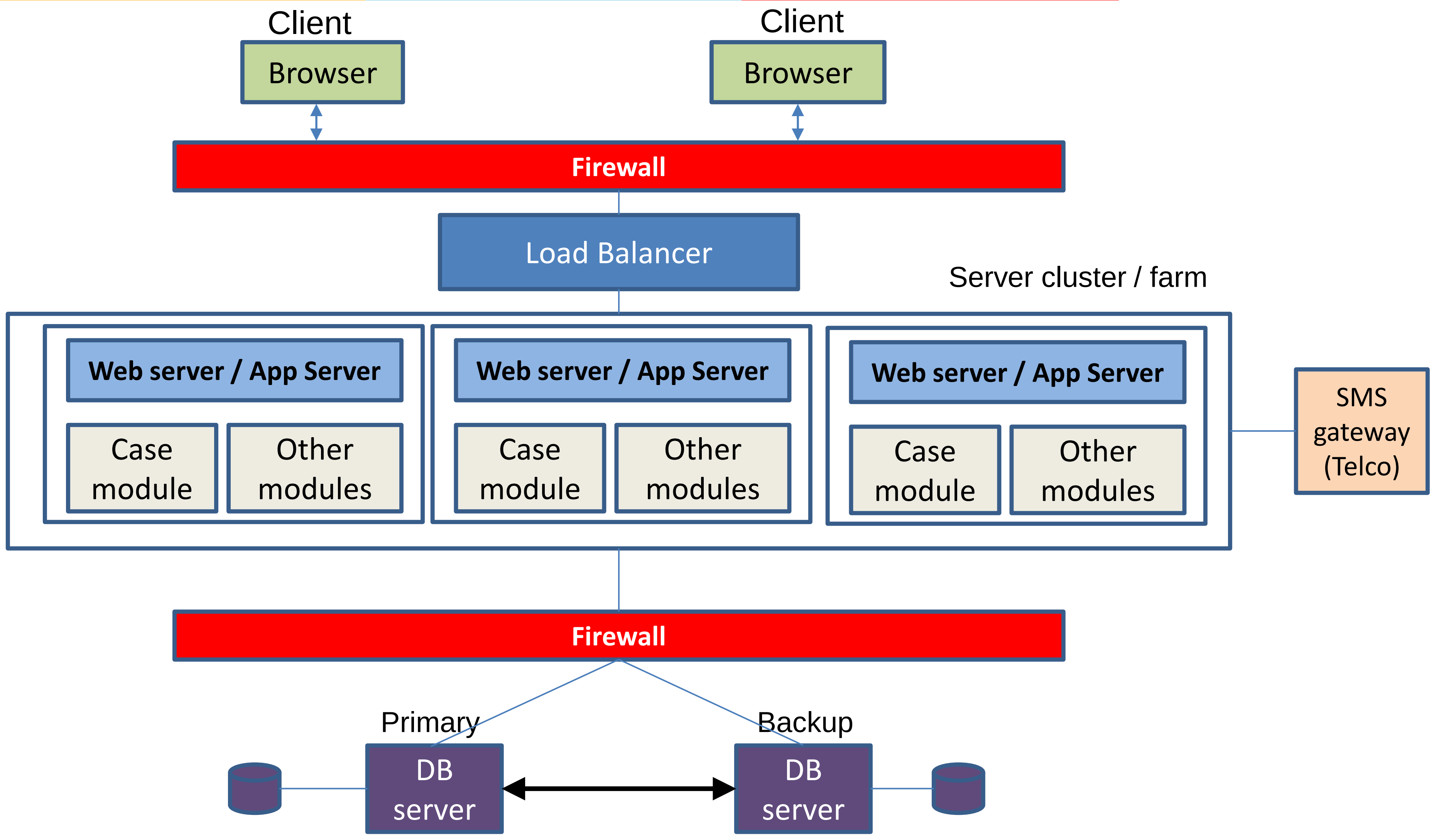
Logical diagram / Component & Connection view

Shows different components and their logical connection with other components



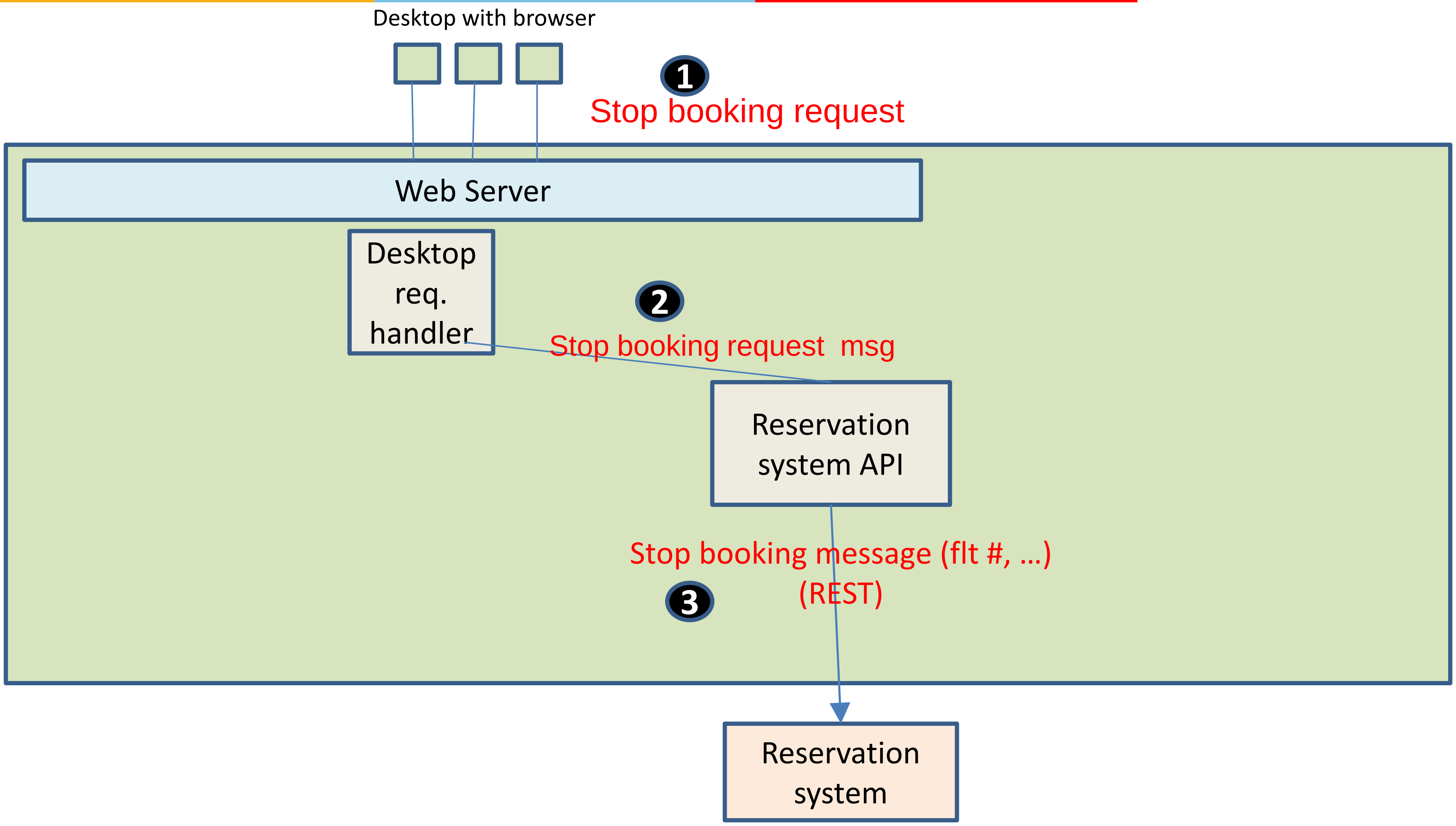
Physical diagram / Deployment diagram

Shows distribution of sw components across computing units



A diagram (view) to explain a scenario:

Ex. Show how the ‘Stop booking’ request works



Software architecture provides design directions

Architecture is like a strategy for software

Marketing strategy of a car company provides directions such as

- What car to manufacture – Luxury or Economy?
- How to market it – What Place, Price and Promotion?

Similarly software architecture provides design directions such as:

- Micro services architecture or Shared data architecture?
- Synchronous communication or Asynchronous?
- Centralized system or distributed system?

Example of design directions

Airline Loyalty system	Space vehicle
• Web application with 3-layers • Import data from Reservation system every night using FTP	• Distributed system with different modules in each system to prevent failure of complete system • Redundant sensors and trajectory measurement algorithms to address component failure

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)?

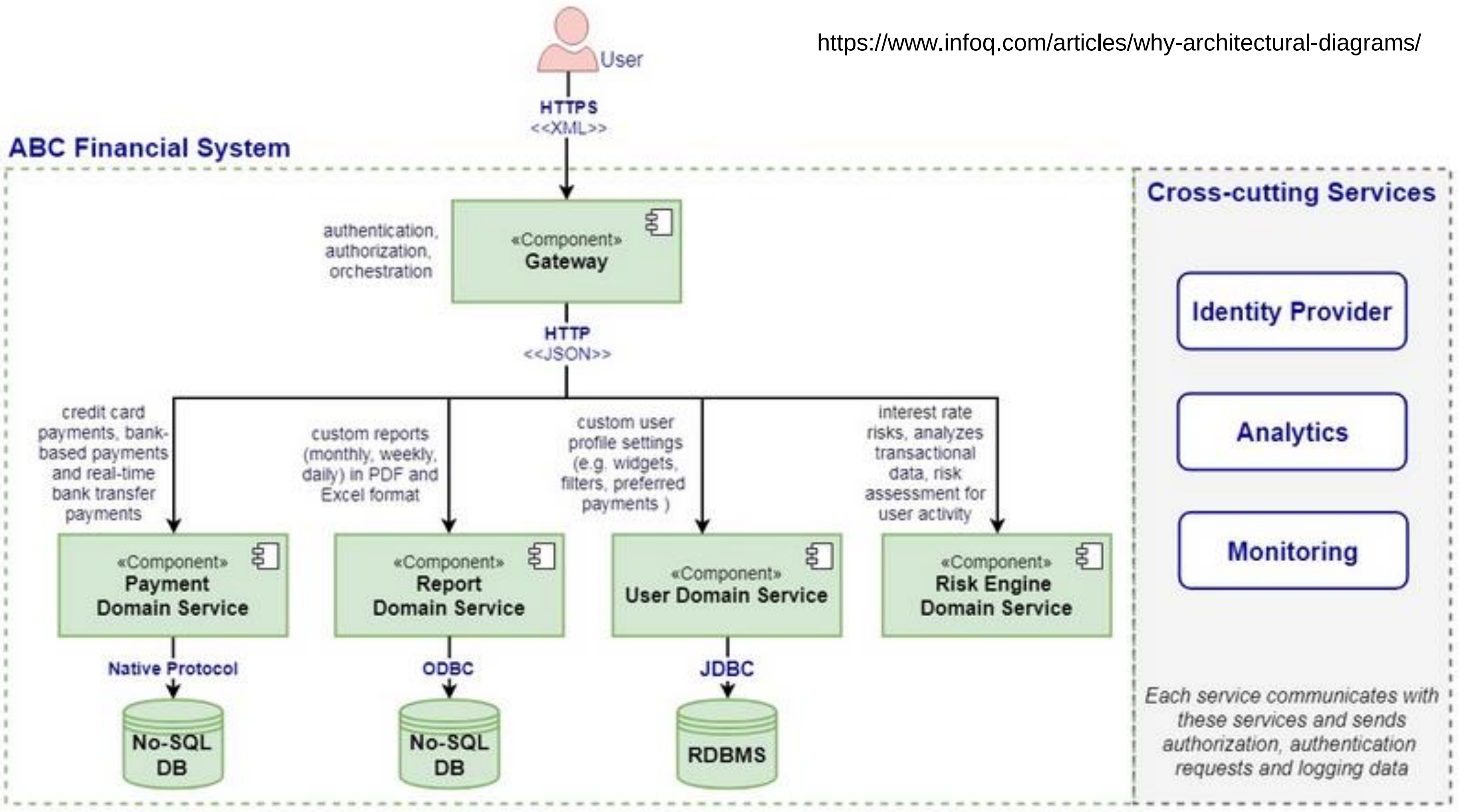
- Helps stakeholders understand how the system will work. Stakeholders are sponsors, developers, operations staff, etc.

Difference between architecture and design

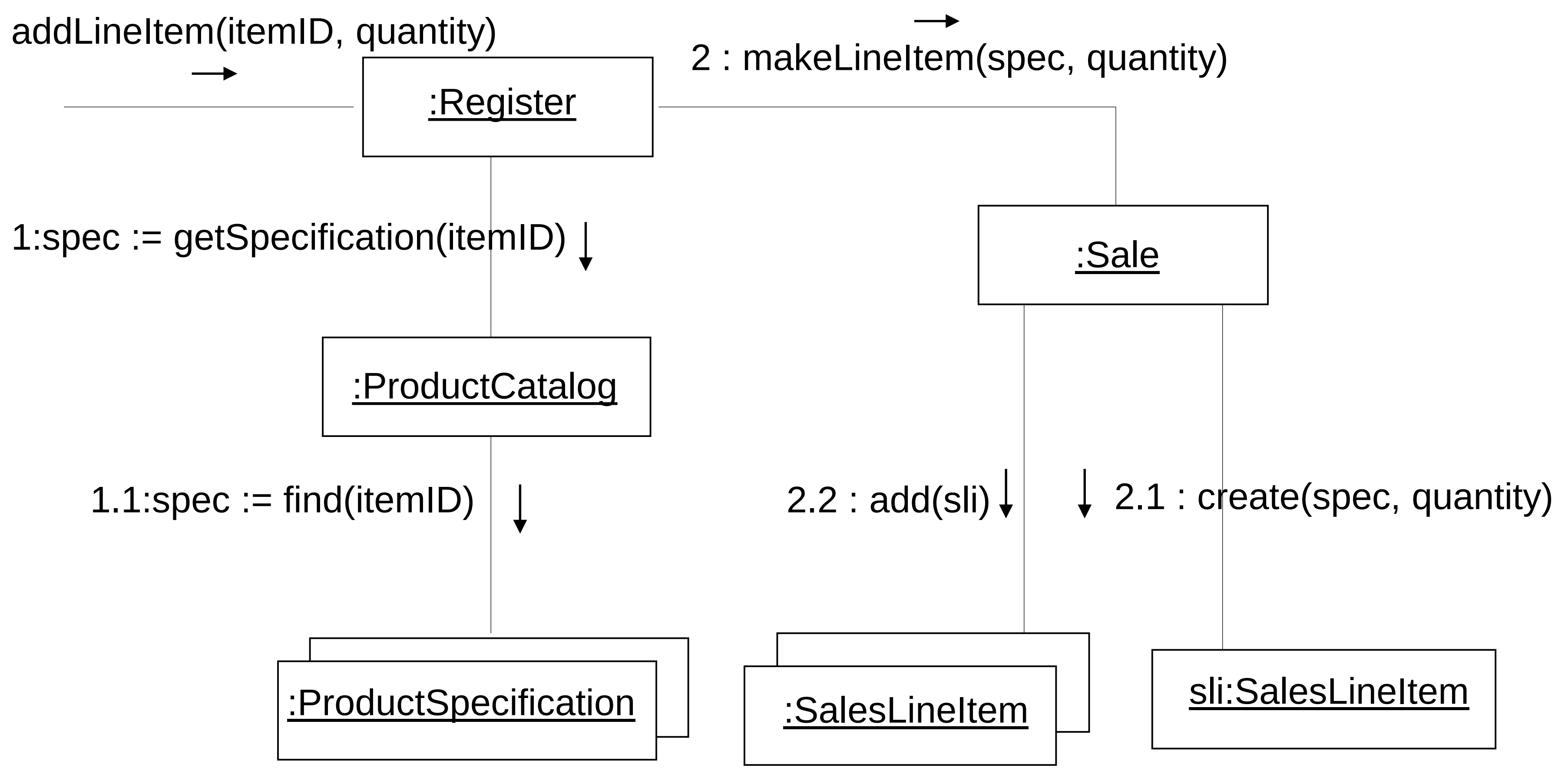
Architecture	Design
Deals with design at a system	Deals with design at Module level
Based on business goals, requirements and constraints	Based on purpose of module
Involves decomposing the system into components and their interactions	Involves designing objects within a module and interactions between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

Example of architecture

https://www.infoq.com/articles/why-architectural-diagrams/


User
HTTPS
<<XML>>
ABC Financial System
authentication, authorization, orchestration
«Component»
Gateway
HTTP
<<JSON>>
credit card payments, bank-based payments and real-time bank transfer payments
custom reports (monthly, weekly, daily) in PDF and Excel format
custom user profile settings (e.g. widgets, filters, preferred payments)
interest rate risks, analyzes transactional data, risk assessment for user activity
«Component»
Payment Domain Service
«Component»
Report Domain Service
«Component»
User Domain Service
«Component»
Risk Engine Domain Service
Native Protocol
ODBC
JDBC
No-SQL DB
No-SQL DB
RDBMS
Cross-cutting Services
Identity Provider
Analytics
Monitoring
Each service communicates with these services and sends authorization, authentication requests and logging data

Example of design of Point-of-Sale system (partial design)


addLineItem(itemID, quantity)
2 : makeLineItem(spec, quantity)
:Register
1:spec := getSpecification(itemID)
:Sale
:ProductCatalog
1.1:spec := find(itemID)
2.2 : add(sli)
2.1 : create(spec, quantity)
:ProductSpecification
:SalesLineItem
sli:SalesLineItem

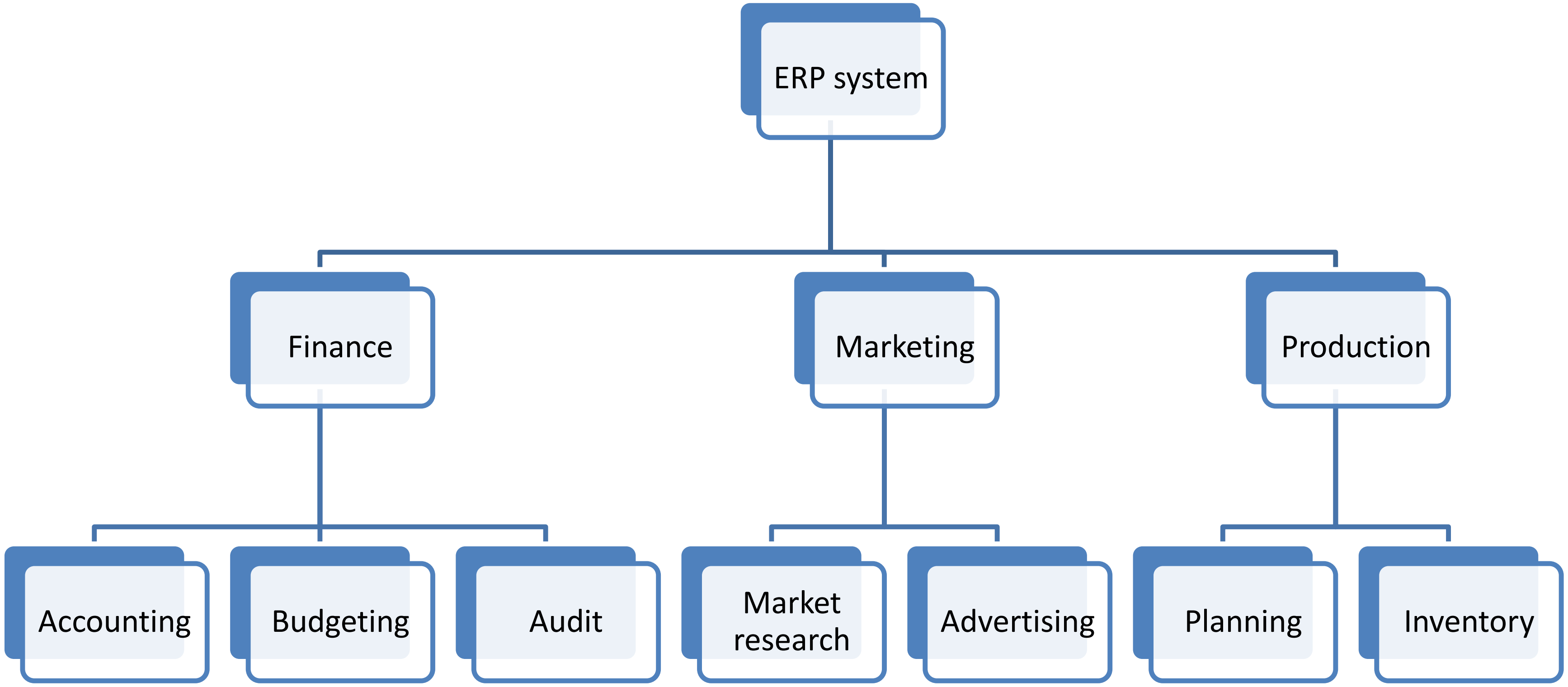
Ref: Book: Applying UML and Patterns by Craig Larman

Architecture structures

- Module structure
- Component & Connection structures
- Deployment structure

Examples of structures

Module decomposition structure



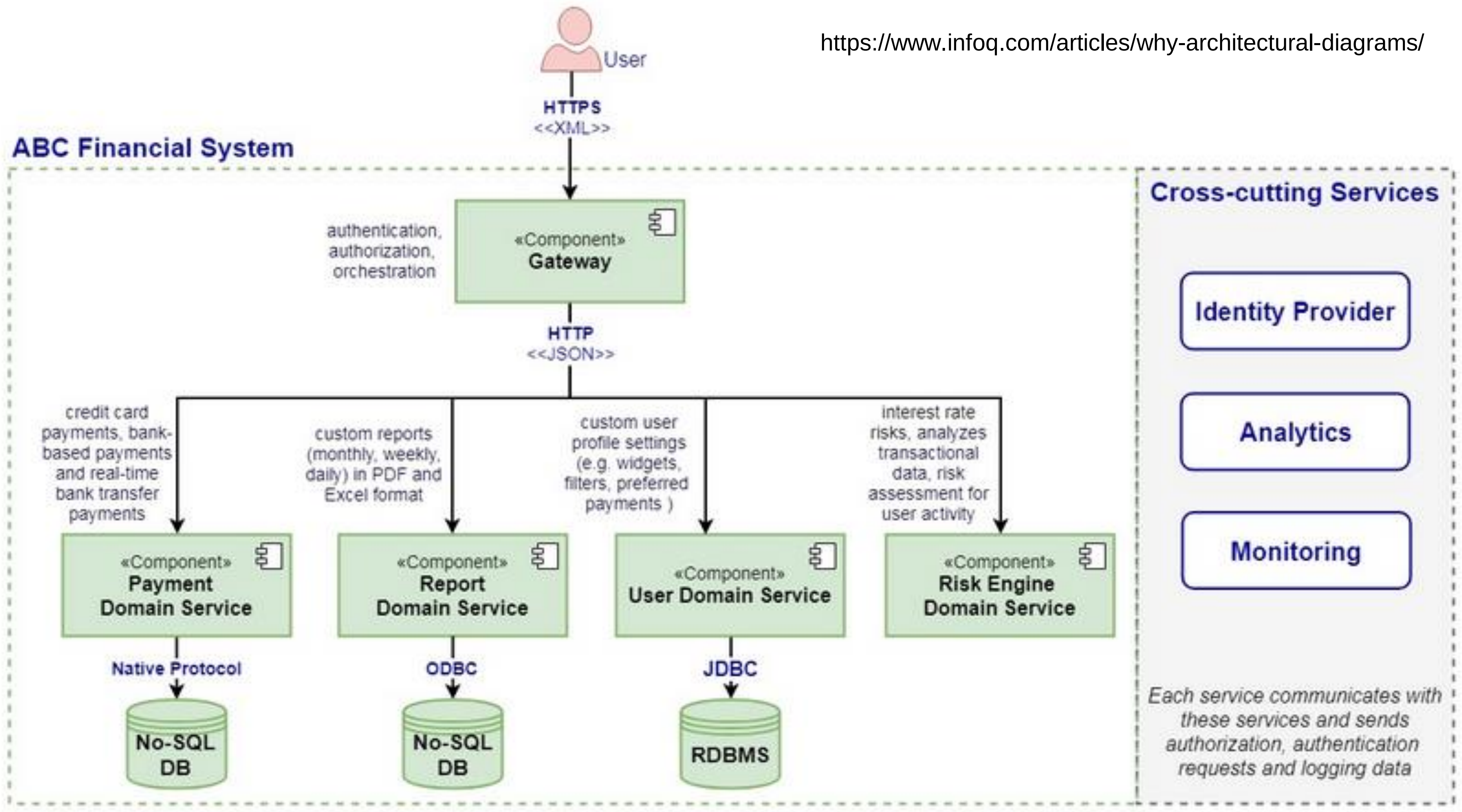
Each is a module

Other module structures

- Package diagram showing modules within a package

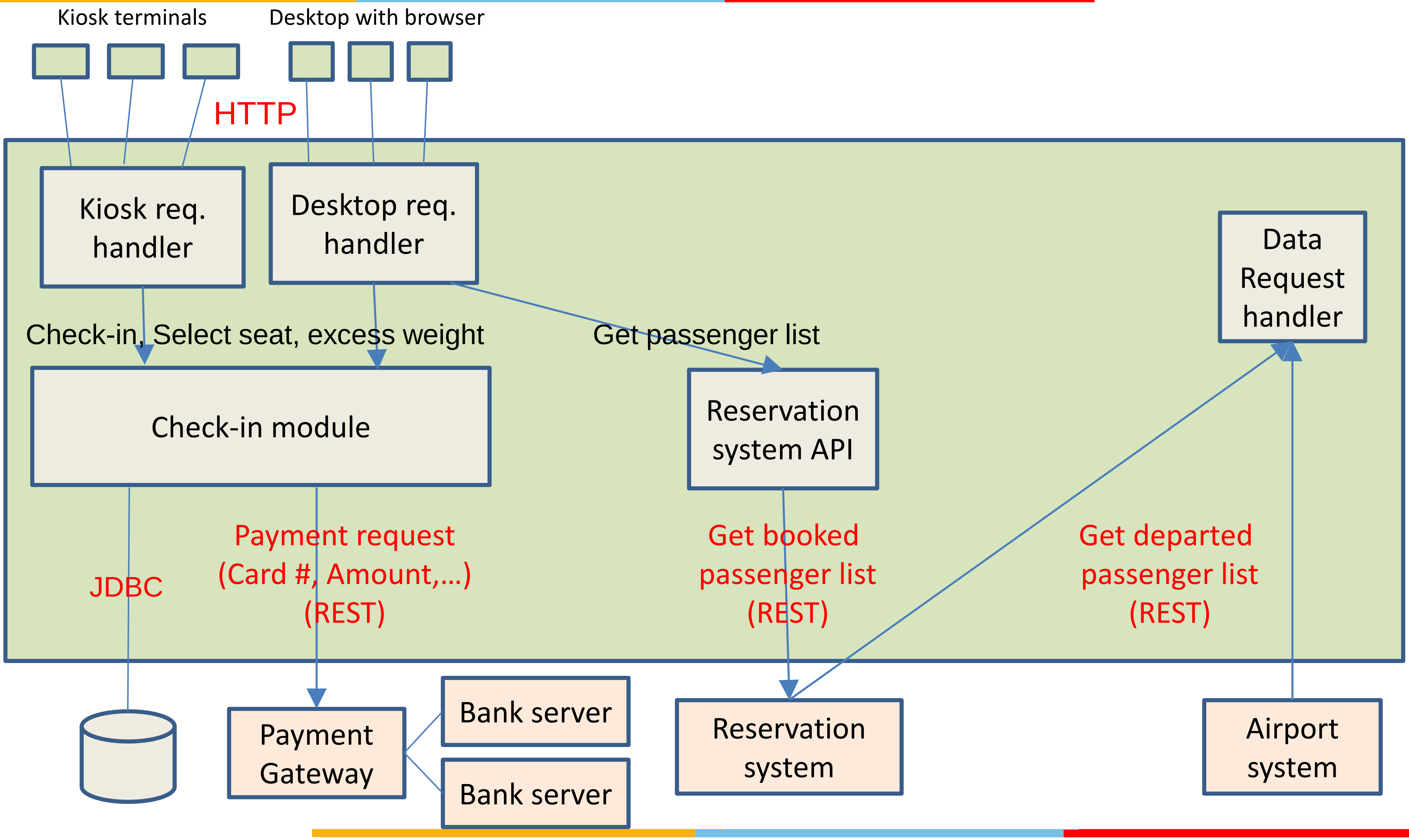
Component & Communication diagram - Example

https://www.infoq.com/articles/why-architectural-diagrams/



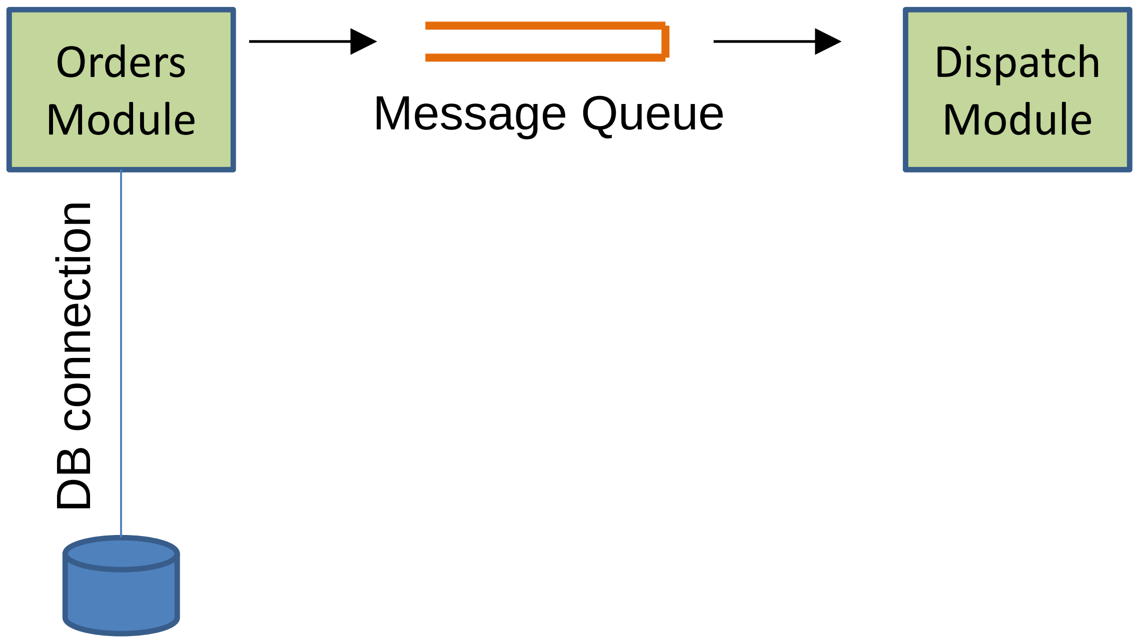
User
HTTPS
<<XML>>
ABC Financial System
authentication, authorization, orchestration
«Component» Gateway
HTTP
<<JSON>>
credit card payments, bank-based payments and real-time bank transfer payments
custom reports (monthly, weekly, daily) in PDF and Excel format
custom user profile settings (e.g. widgets, filters, preferred payments)
interest rate risks, analyzes transactional data, risk assessment for user activity
«Component» Payment Domain Service
«Component» Report Domain Service
«Component» User Domain Service
«Component» Risk Engine Domain Service
Native Protocol
ODBC
JDBC
No-SQL DB
No-SQL DB
RDBMS
Cross-cutting Services
Identity Provider
Analytics
Monitoring
Each service communicates with these services and sends authorization, authentication requests and logging data

Component & Connection diagram



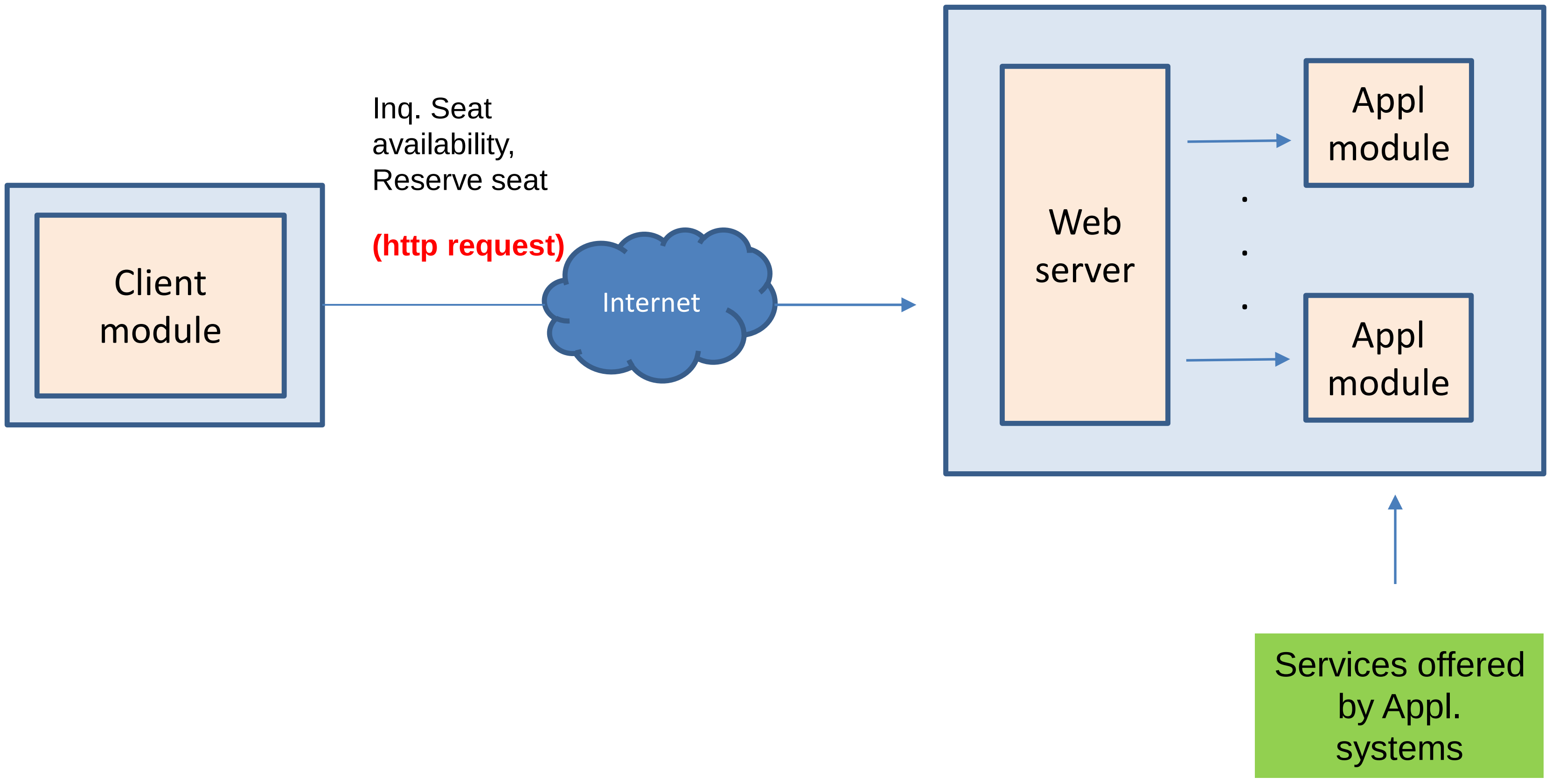
Connection mechanisms

- Message queues
- Shared memory
- Database connections
- Representational State Transfer (REST)

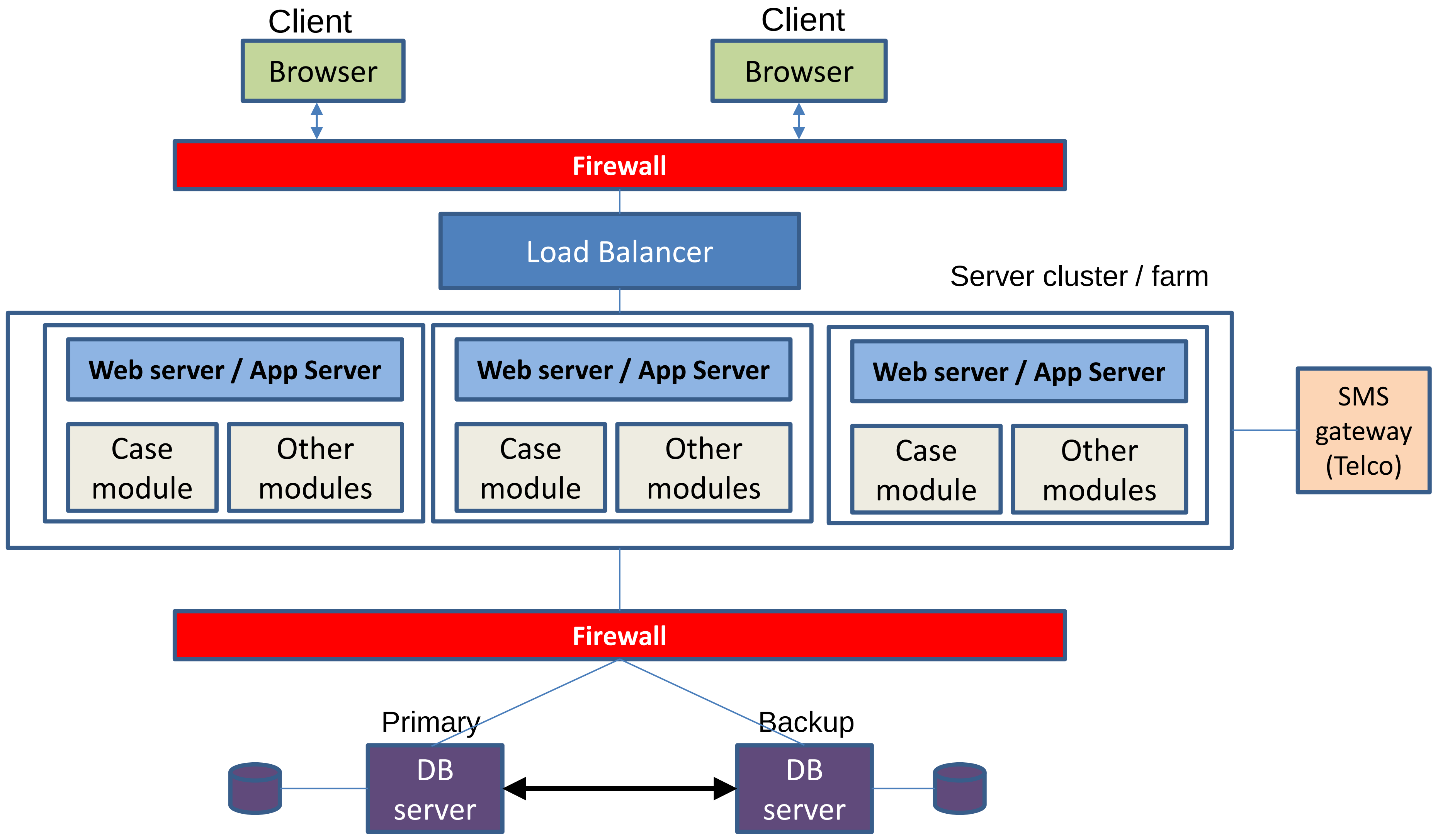


Connection mechanisms

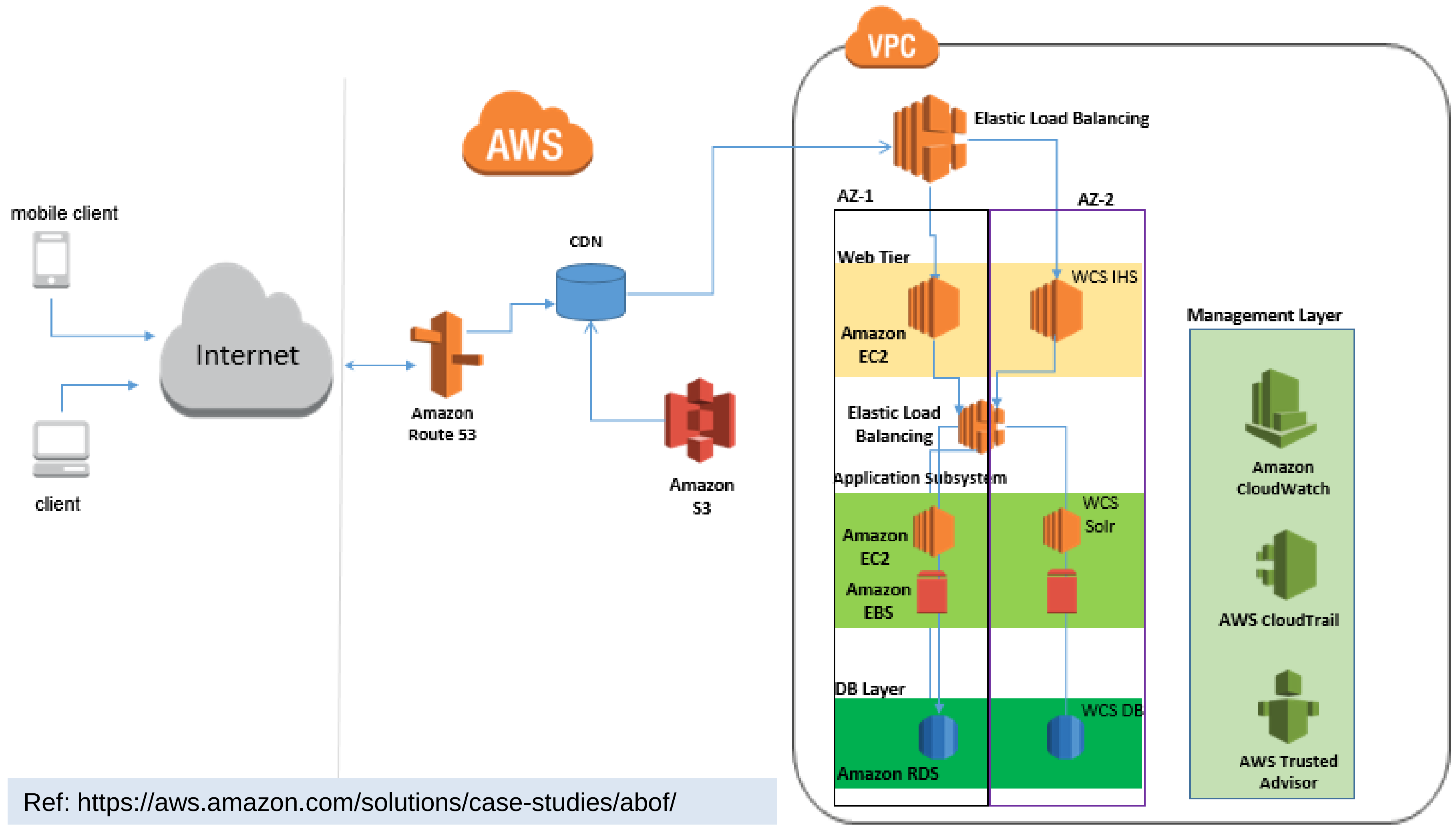
- REST



Deployment view: Example



Deployment diagram - Example



Ref: https://aws.amazon.com/solutions/case-studies/abof/

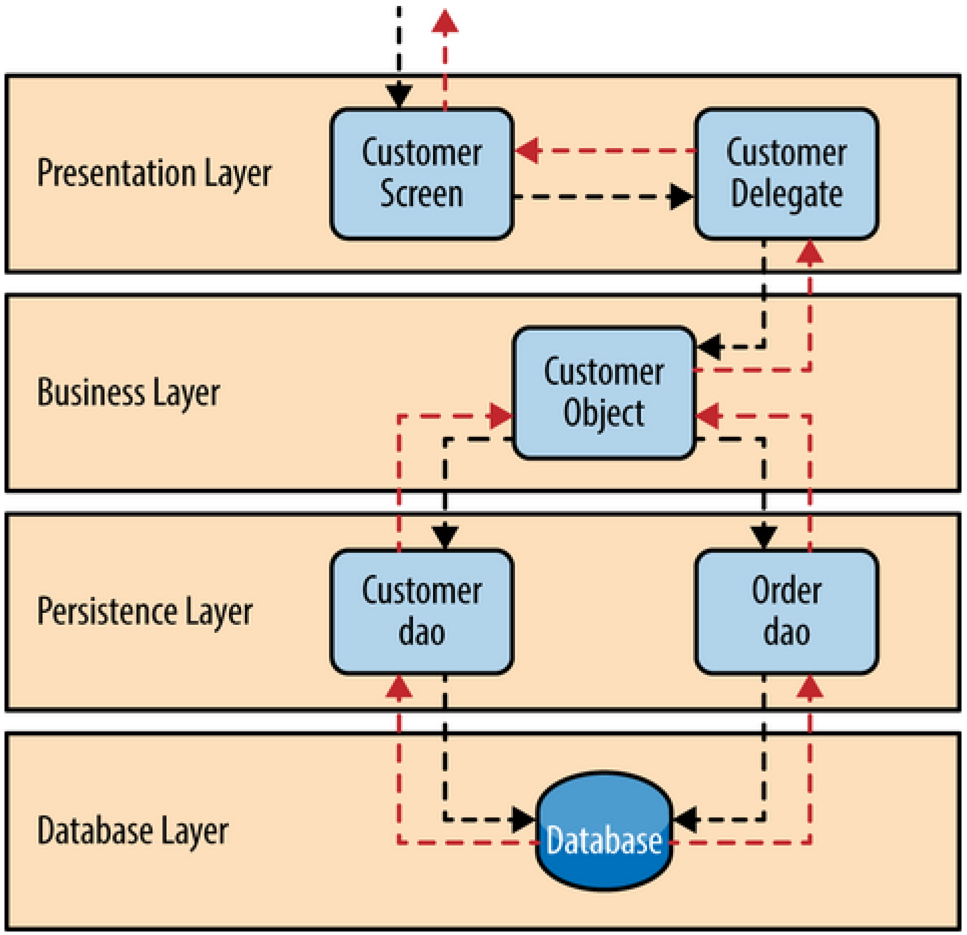
Architecture patterns

- Layered pattern
- Publish – Subscribe
- Pipeline pattern
- Etc.

Examples of patterns

innovate achieve lead

Layered Pattern - Example

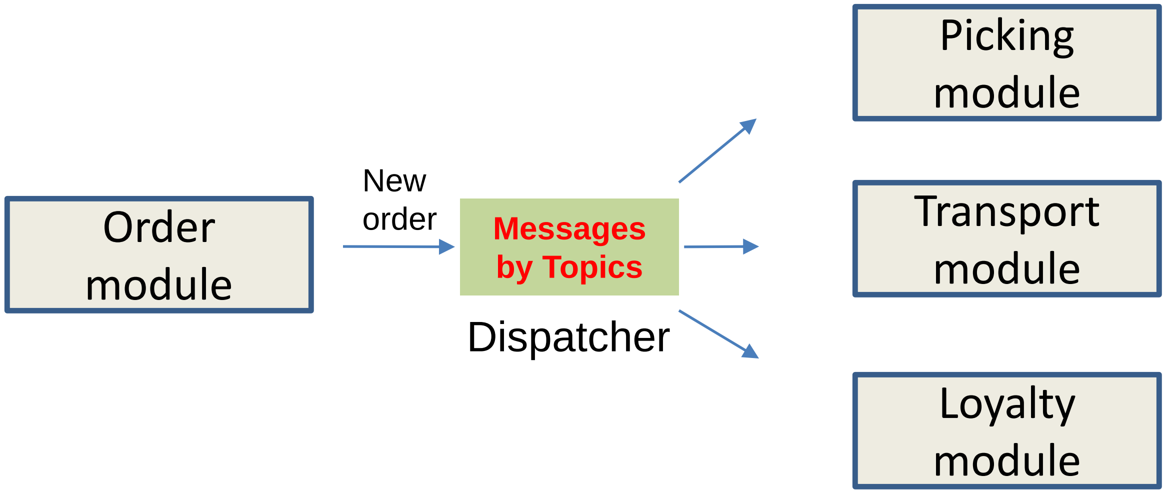


https://www.oreilly.com/ideas/software-architecture-patterns/page/2/layered-architecture

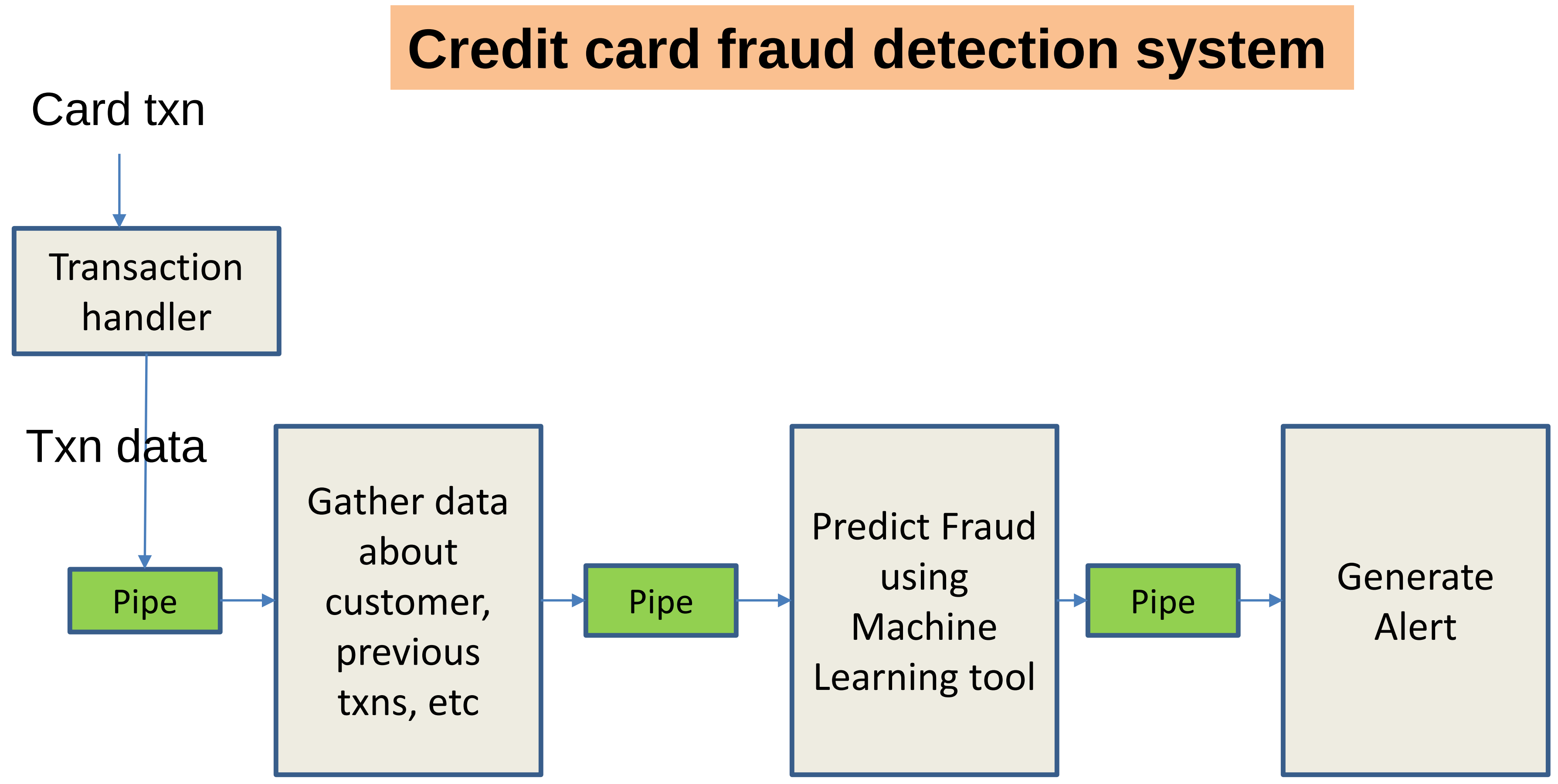
Publish - Subscribe

Example

- Order creation event is subscribed to by Picking module, Transport module, Loyalty module



Pipeline


Credit card fraud detection system
Card txn
Transaction handler
Txn data
Pipe
Gather data about customer, previous txns, etc
Pipe
Predict Fraud using Machine Learning tool
Pipe
Generate Alert

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.

- Non-functional requirements (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

Examples of Functional requirements by module

Products

- Update inventory
- Define discount

Orders:

- Create new order
- Mark an order as shipped

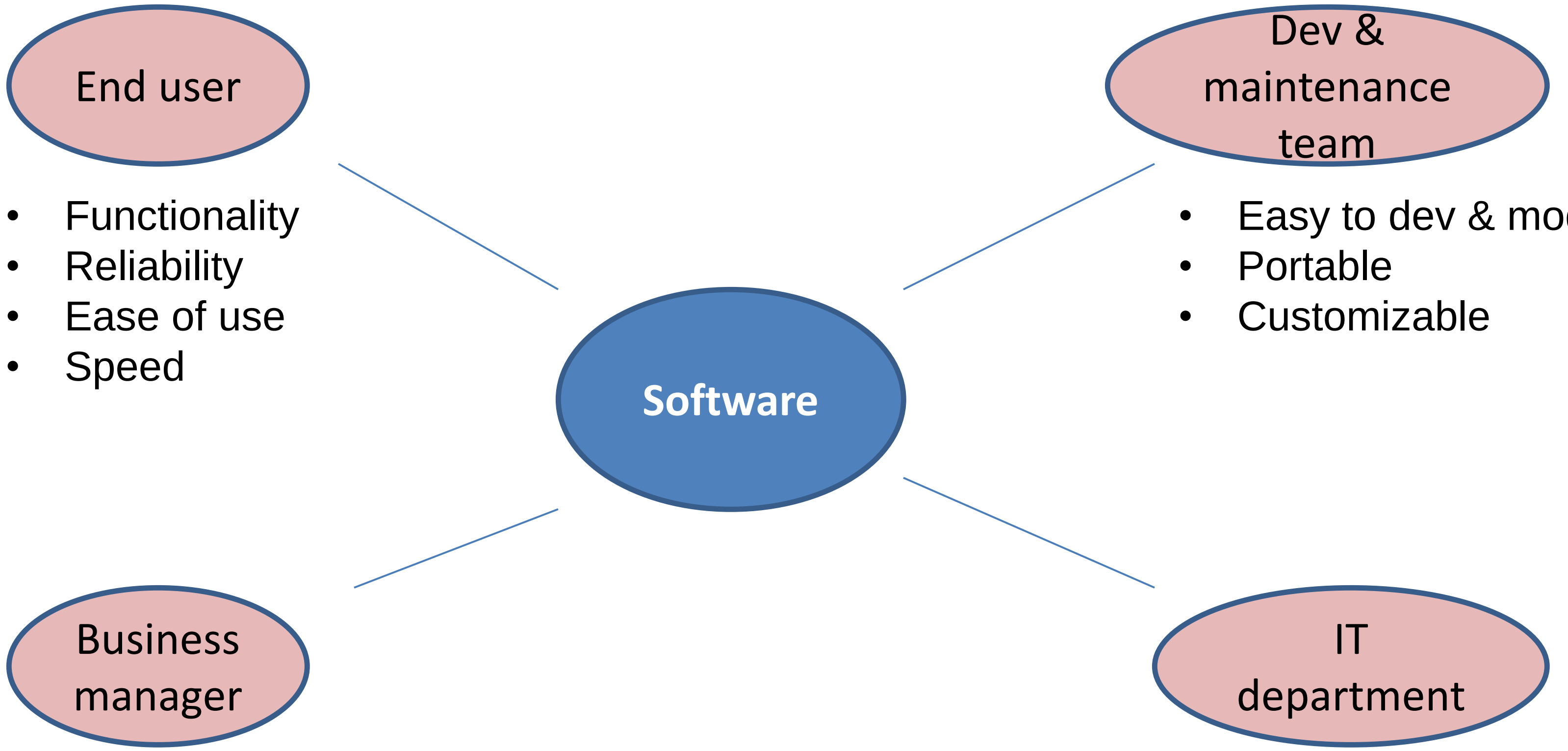
Complaints

- Create new complaint
- Identify frequently occurring complaints

Examples of non-functional requirements

- Response time should be less than 3 seconds for 95% of transactions
- The system up-time should be greater than 99.9999%
- Salary data should be accessible to employee, manager and HR department only
- Adding a new payment gateway should be easy

Expectations of different stakeholders



End user

- Functionality
- Reliability
- Ease of use
- Speed

Dev & maintenance team

- Easy to dev & modify
- Portable
- Customizable

Business manager

- Functionality
- Low cost
- Short time to market

IT department

- Ease of installation
- Plug & play
- Scalability

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

Challenges in software architecture

Trade-offs: Trying to satisfy one requirement may compromise another

- Example:
- Trying to strengthen security may degrade performance
- Trying to improve performance may compromise modularity & maintainability

Requirements keep changing, especially in case of a new product

Ensuring that development conforms to architecture principles – layers, common code, info hiding

Role of an architect

Role of an architect

1. Understand the business goals
2. Identify Architecturally significant requirements
3. Design an appropriate architecture for the system
4. Explain the architecture to stakeholders and show how it meets business goals & requirements
5. Guide the software development team

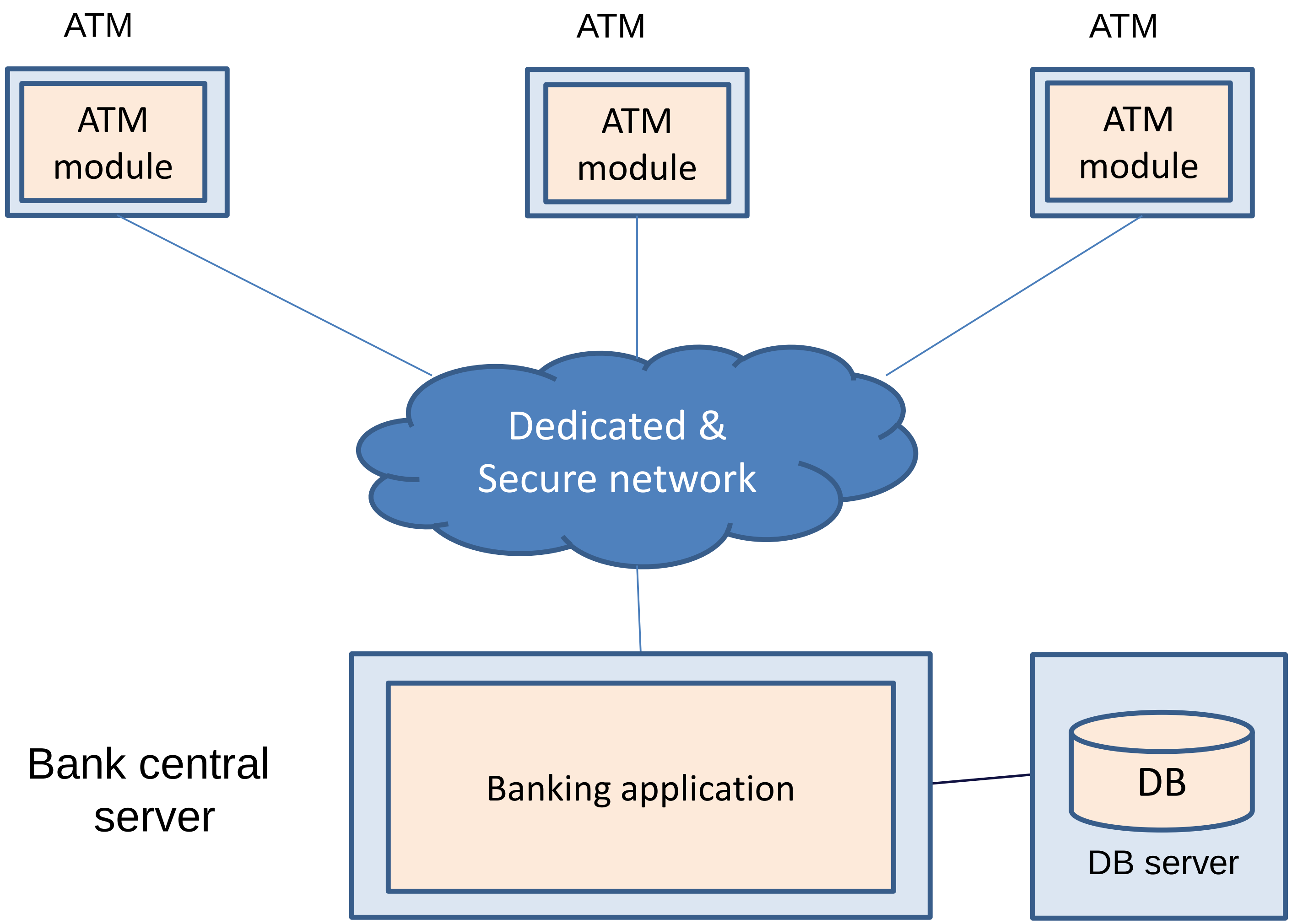
Exercise: Draw architecture of a Retail banking system with ATMs

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

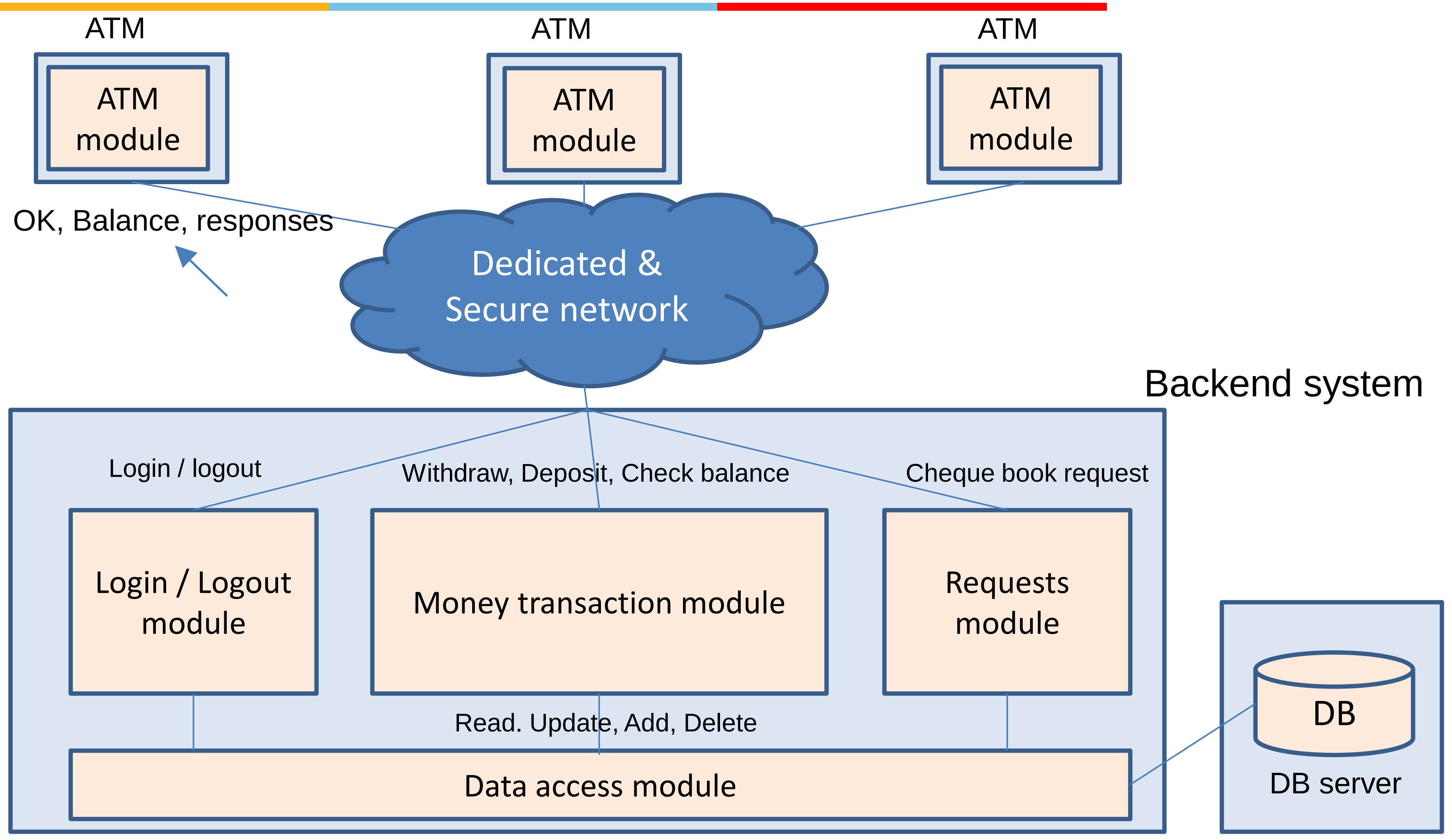
Architecture of a banking software supporting ATM

See next slide

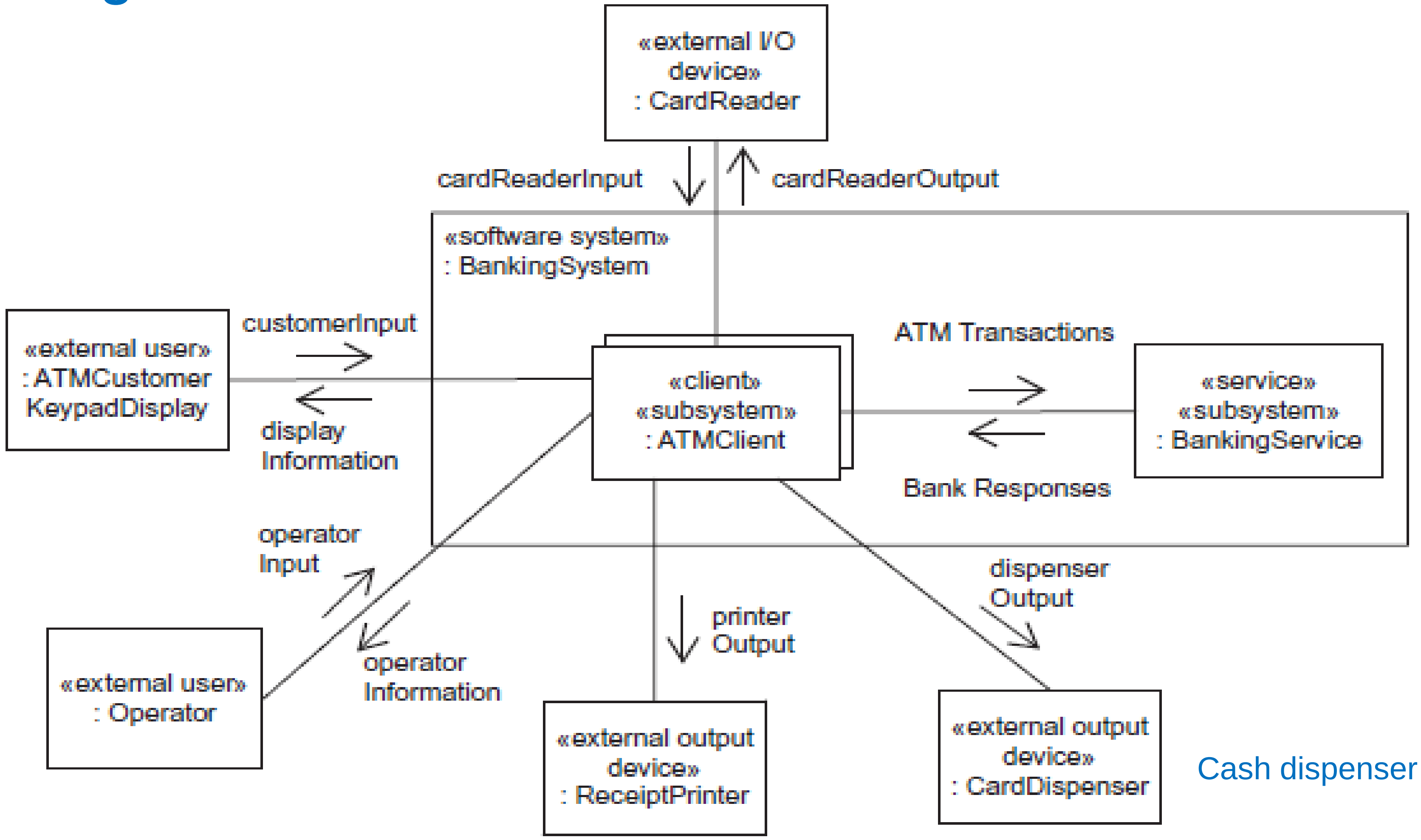
High level diagram



Component and Connection diagram



Message communications


«external I/O device» : CardReader
cardReaderInput
cardReaderOutput
«software system» : BankingSystem
«external user» : ATMCustomer KeypadDisplay
customerInput
display Information
«client» «subsystem» : ATMClient
ATM Transactions
Bank Responses
«service» «subsystem» : BankingService
operator Input
operator Information
«external user» : Operator
printer Output
dispenser Output
«external output device» : ReceiptPrinter
«external output device» : CardDispenser

Cash dispenser

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

From book: Software Modelling & design by Hassan Gomaa

Objective of the course

To equip you with skills to be an effective architect

Learning outcomes

1. Knowledge of different types of system requirements (quality attributes) and tactics to achieve them
2. Be able to identify architecturally significant requirements
3. Be able to architect the system & document it
4. Evaluate a given architecture
5. Possess knowledge of architecture patterns and their applications
6. Be aware of different technologies & techniques such as
   - NoSQL
   - Big data Analytics
   - Mobile apps
   - Machine Learning
   - Block chain
   - IoT
7. Be able to leverage Cloud services to architect systems

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)

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 your system, 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 innovate achieve lead

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

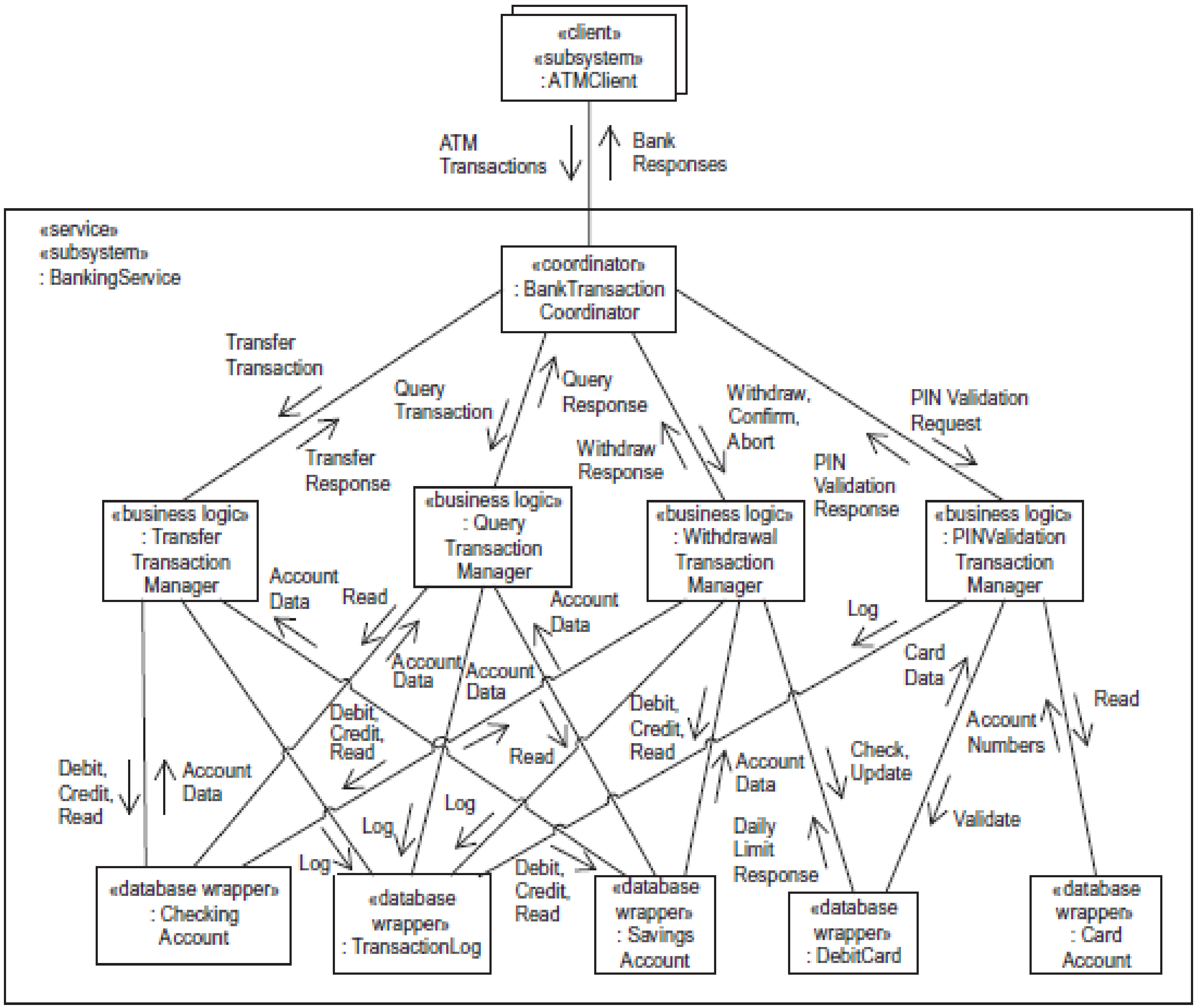

«client» «subsystem» : ATMClient
ATM Transactions
Bank Responses
«service» «subsystem» : BankingService
«coordinator» : BankTransaction Coordinator
Transfer Transaction
Transfer Response
Query Transaction
Query Response
Withdraw Response
Withdraw, Confirm, Abort
PIN Validation Request
PIN Validation Response
«business logic» : Transfer Transaction Manager
«business logic» : Query Transaction Manager
«business logic» : Withdrawal Transaction Manager
«business logic» : PINValidation Transaction Manager
Account Data
Read
Account Data
Account Data
Account Data
Log
Card Data
Read
Account Numbers
Debit, Credit, Read
Read
Debit, Credit, Read
Account Data
Check, Update
Debit, Credit, Read
Account Data
Validate
Log
Log
Log
Daily Limit Response
Debit, Credit, Read
«database wrapper» : Checking Account
«database wrapper» : TransactionLog
«database wrapper» : Savings Account
«database wrapper» : DebitCard
«database wrapper» : Card Account

Figure 21.26. Integrated communication diagram for Banking Service subsystem

From book: Software Modelling & design by Hassan Gomaa

Real time systems – Automated guided vehicle

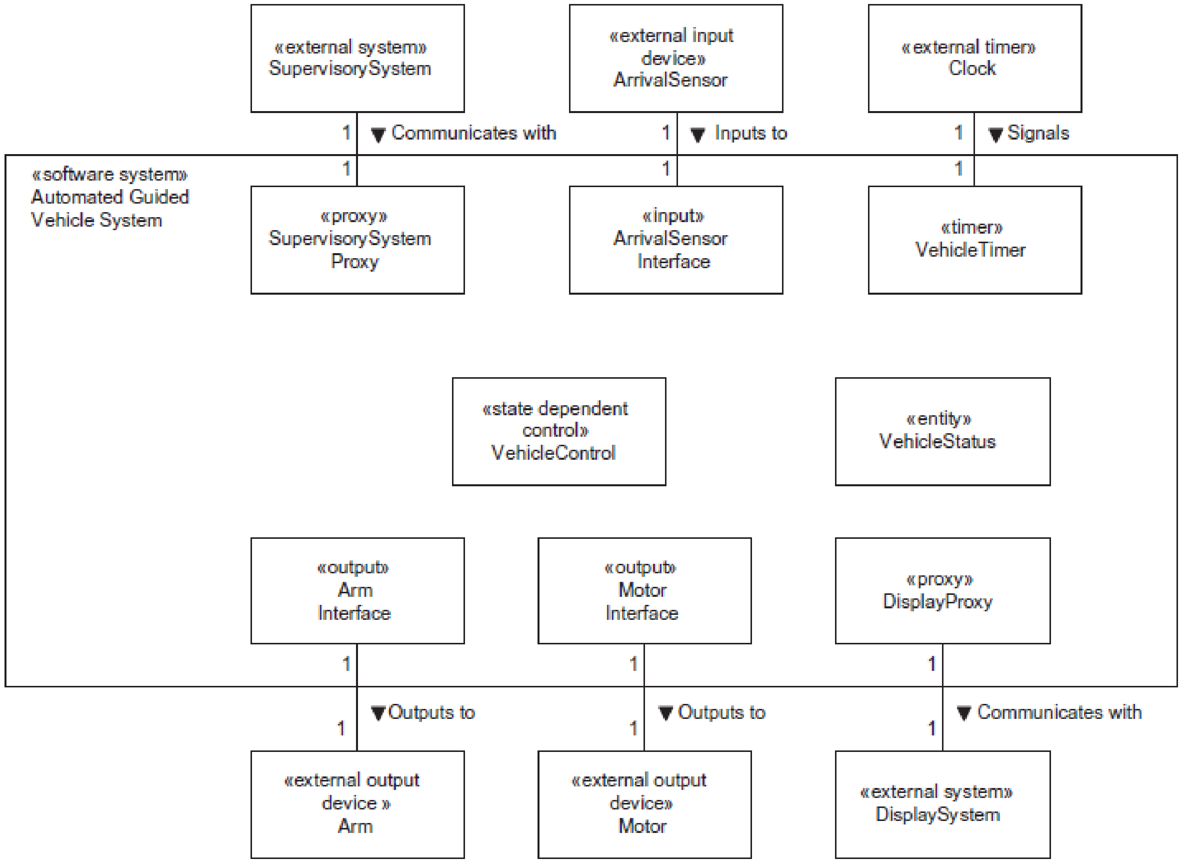

«external system» SupervisorySystem
«external input device» ArrivalSensor
«external timer» Clock
1
▼ Communicates with
1
▼ Inputs to
1
▼ Signals
«software system» Automated Guided Vehicle System
1
«proxy» SupervisorySystem Proxy
1
«input» ArrivalSensor Interface
1
«timer» VehicleTimer
«state dependent control» VehicleControl
«entity» VehicleStatus
«output» Arm Interface
«output» Motor Interface
«proxy» DisplayProxy
1
▼ Outputs to
1
1
▼ Outputs to
1
1
▼ Communicates with
1
«external output device » Arm
«external output device» Motor
«external system» DisplaySystem

Figure 24.4. Object structuring for the Automated Guided Vehicle System

Real time system

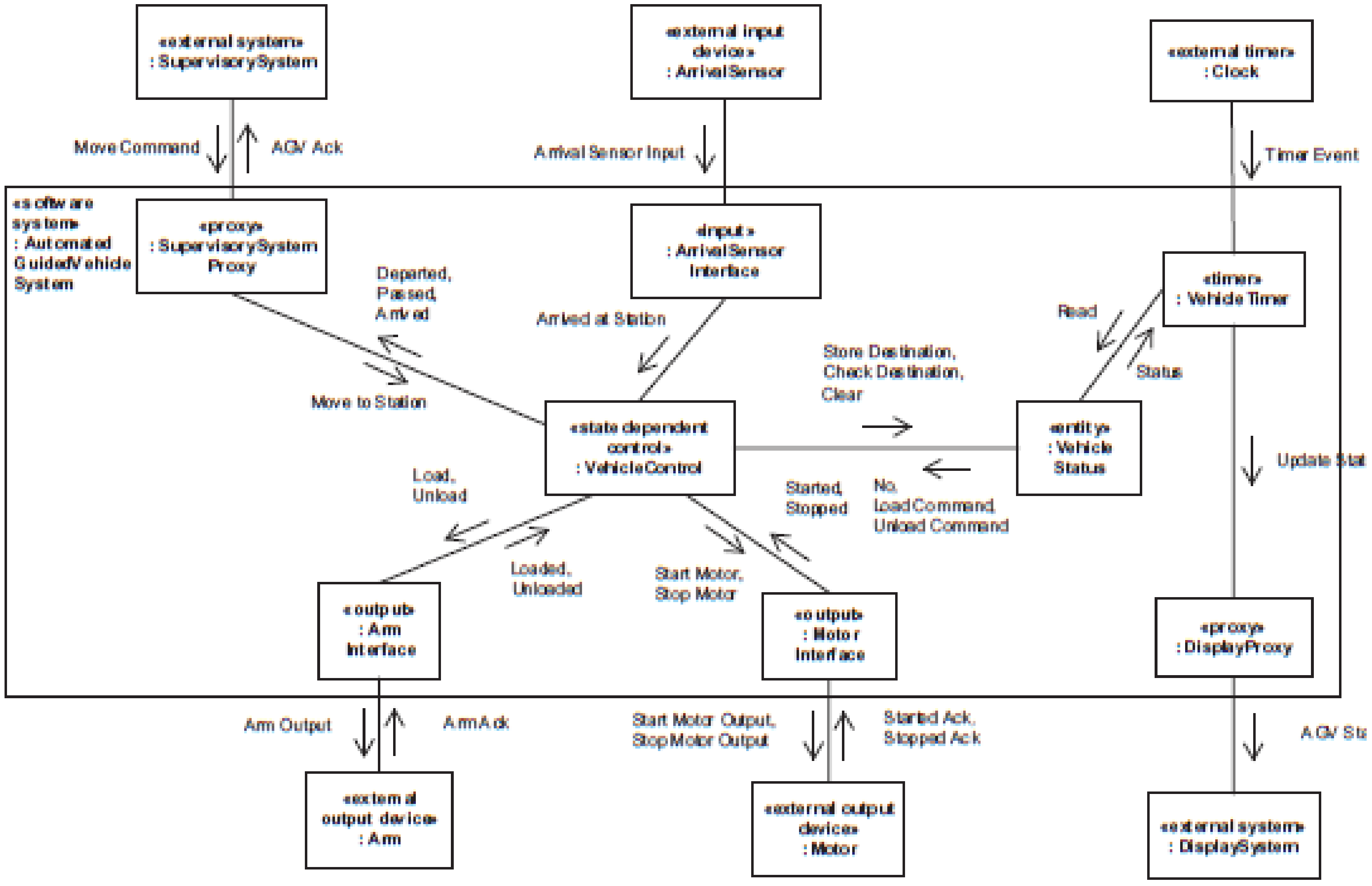

«external system» : SupervisorySystem
«external input device» : ArrivalSensor
«external timer» : Clock
Move Command
AGV Ack
Arrival Sensor Input
Timer Event
«software system» : AutomatedGuidedVehicleSystem
«proxy» : SupervisorySystemProxy
«input» : ArrivalSensorInterface
«timer» : VehicleTimer
Departed, Passed, Arrived
Arrived at Station
Read
Status
Move to Station
Store Destination, Check Destination, Clear
«state dependent control» : VehicleControl
«entity» : VehicleStatus
Update Stat
Load, Unload
Started, Stopped
No, Load Command, Unload Command
Loaded, Unloaded
Start Motor, Stop Motor
«output» : ArmInterface
«output» : MotorInterface
«proxy» : DisplayProxy
Arm Output
Arm Ack
Start Motor Output, Stop Motor Output
Started Ack, Stopped Ack
AGV Sta
«external output device» : Arm
«external output device» : Motor
«external system» : DisplaySystem

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

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

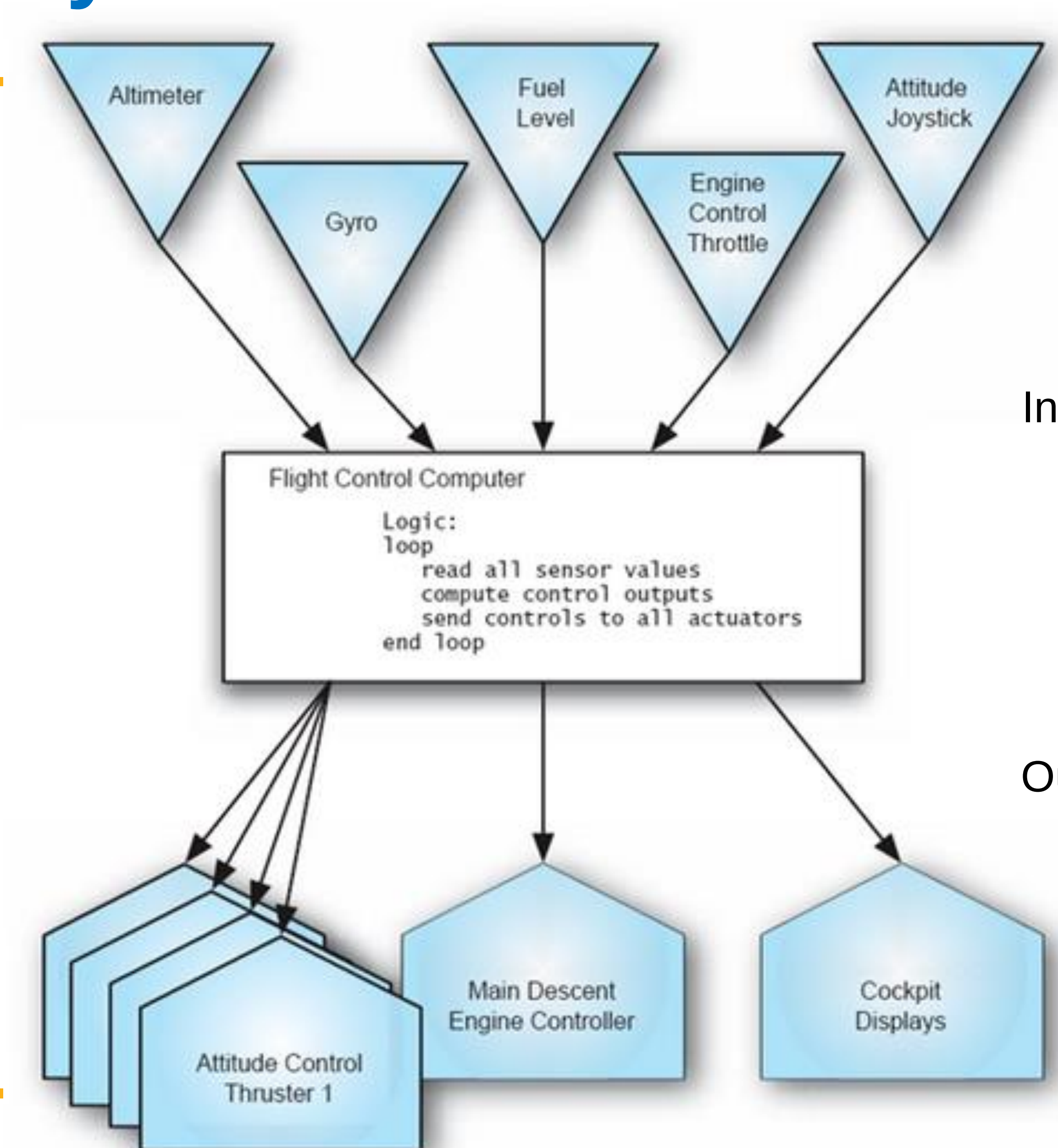
Type of application	Expectations
Enterprise apps – Banking, Telecom, Airline, Retail	Reliability, security, performance
Industrial control, avionics	Time critical, very reliable, life threatening if it fails
Operating system	Reliable, support different devices, high performance, muti-processing, security
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

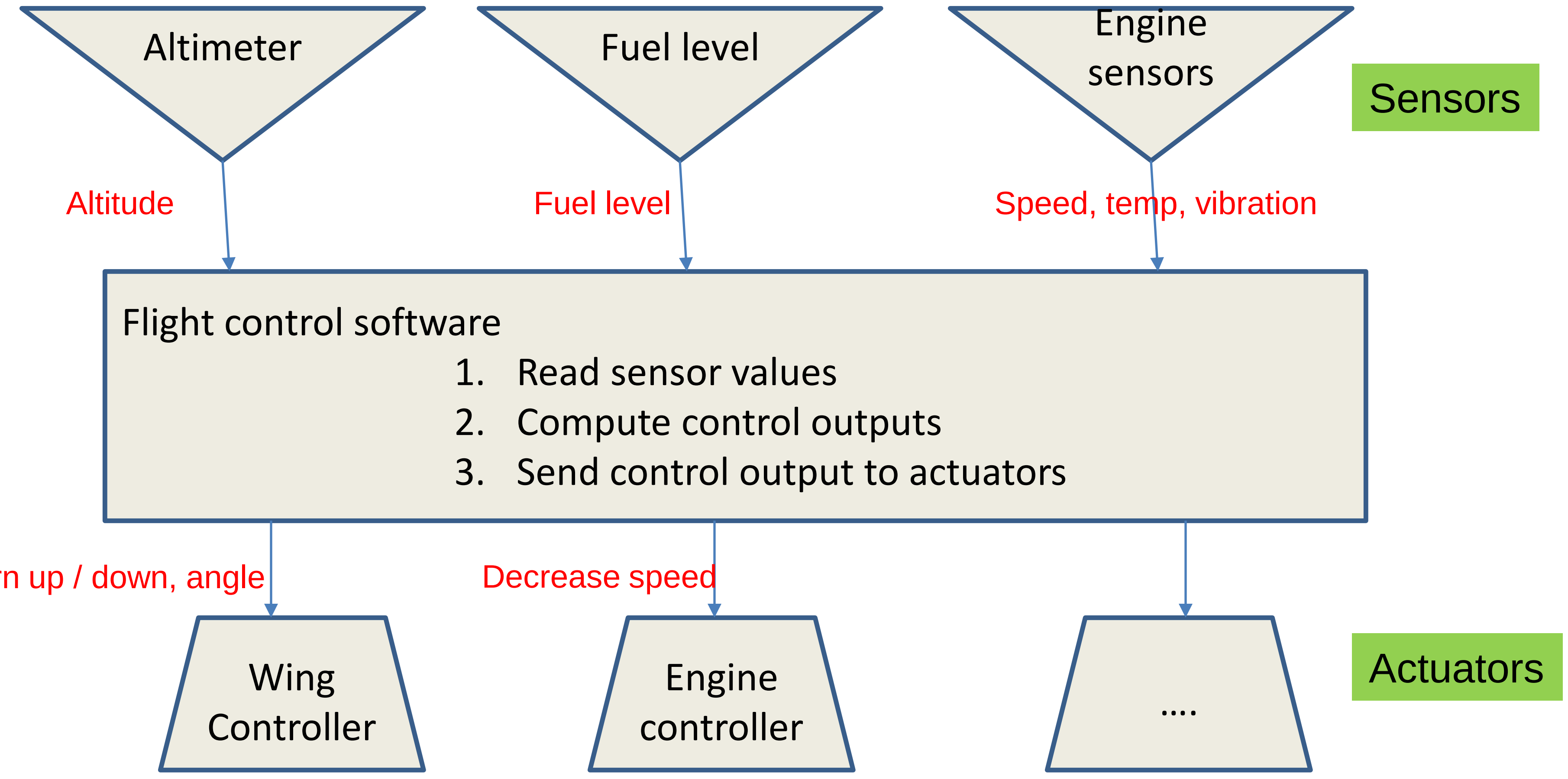


```
Logic:
loop
    read all sensor values
    compute control outputs
    send controls to all actuators
end loop
```

Inputs

Outputs

Real time software architecture

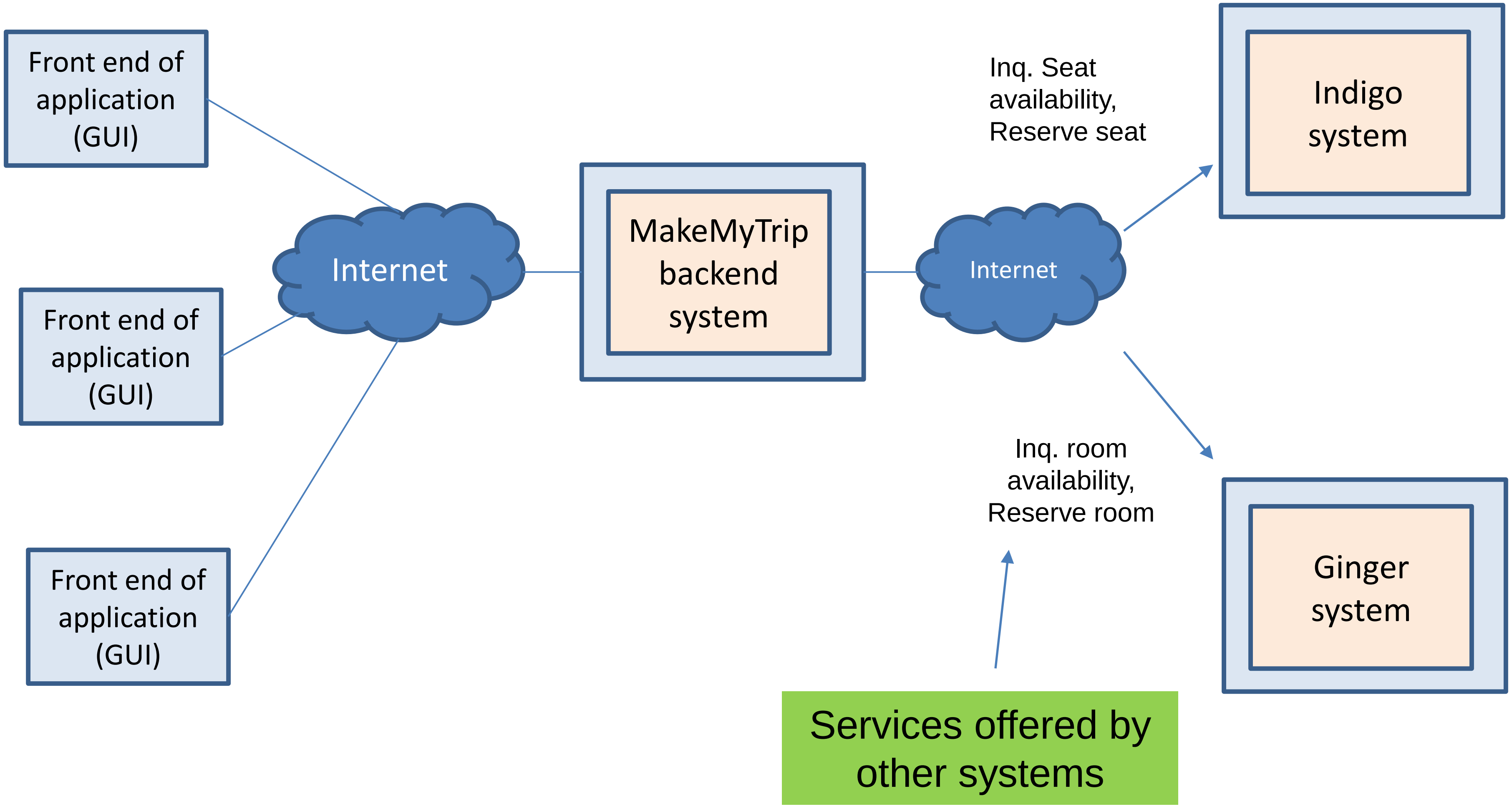


Flight control software

1. Read sensor values
2. Compute control outputs
3. Send control output to actuators

Service Oriented Architecture

Example: MakeMyTrip.com


Front end of application (GUI)
Front end of application (GUI)
Front end of application (GUI)
Internet
MakeMyTrip backend system
Internet
Inq. Seat availability, Reserve seat
Indigo system
Inq. room availability, Reserve room
Ginger system
Services offered by other systems

Layered pattern:
Example: Architecture of an Information system

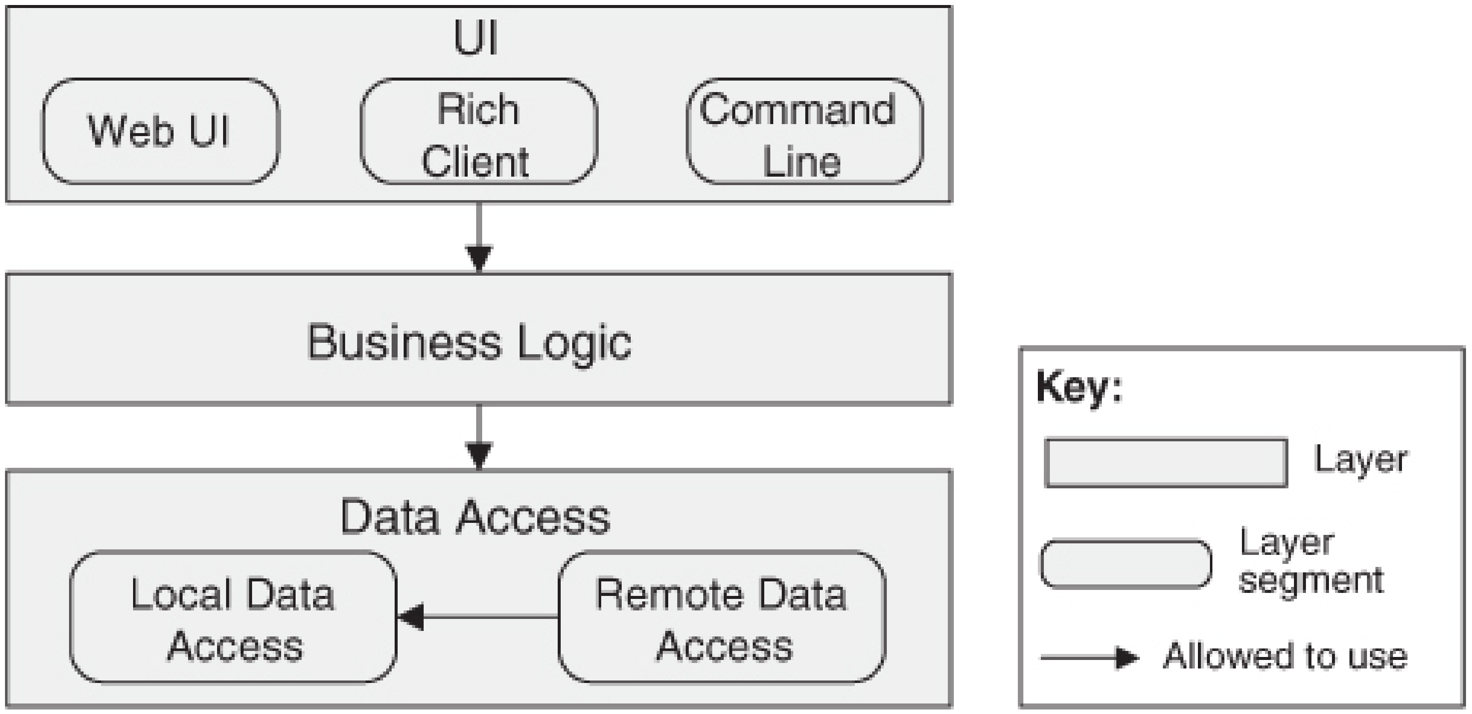


Figure 13.5. Layered design with segmented layers

Ref: Software Architecture in Practice by Len Bass and others