Architecture Design Process:

The Attribute-Driven Design Method

Lotfi ben Othmane

Expected Outcomes

1. Understand what a software architecture is and explain why it is important
2. Understand the relationship between software qualities attributes and software architectures
3. Ability to elicit software architecture drivers
4. Ability to use architecture styles, patterns, and tactics
5. Ability to use the attribute-driven method to design software architecture
6. Ability to document software architecture
7. Ability to evaluate software architecture

- “A good scientist is a person with original ideas”
- “A good engineer is a person who makes design that works with as few original ideas as possible”

Freeman Dyson

Architecture Design

Architecture drivers

Design purposes

Quality attributes

Primary functionalities

Architectural concerns

Constraints

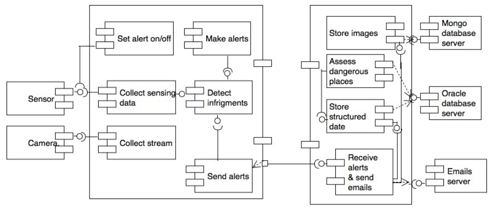
Design concepts

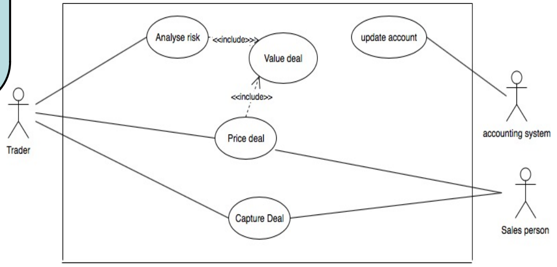
Selects and instantiates

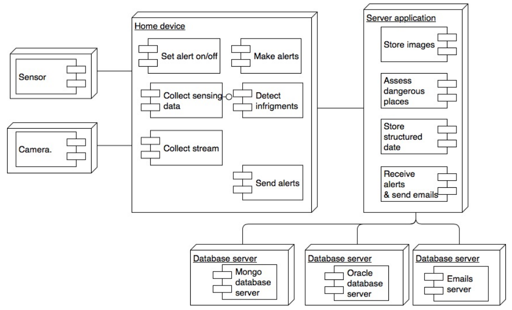
Candidate design decisions

Architect

Architecture structures







Architecture Design

Architecture design is:

making decisions to satisfy project’s goals and constraints.

Architecture Design

A decision is architectural if:

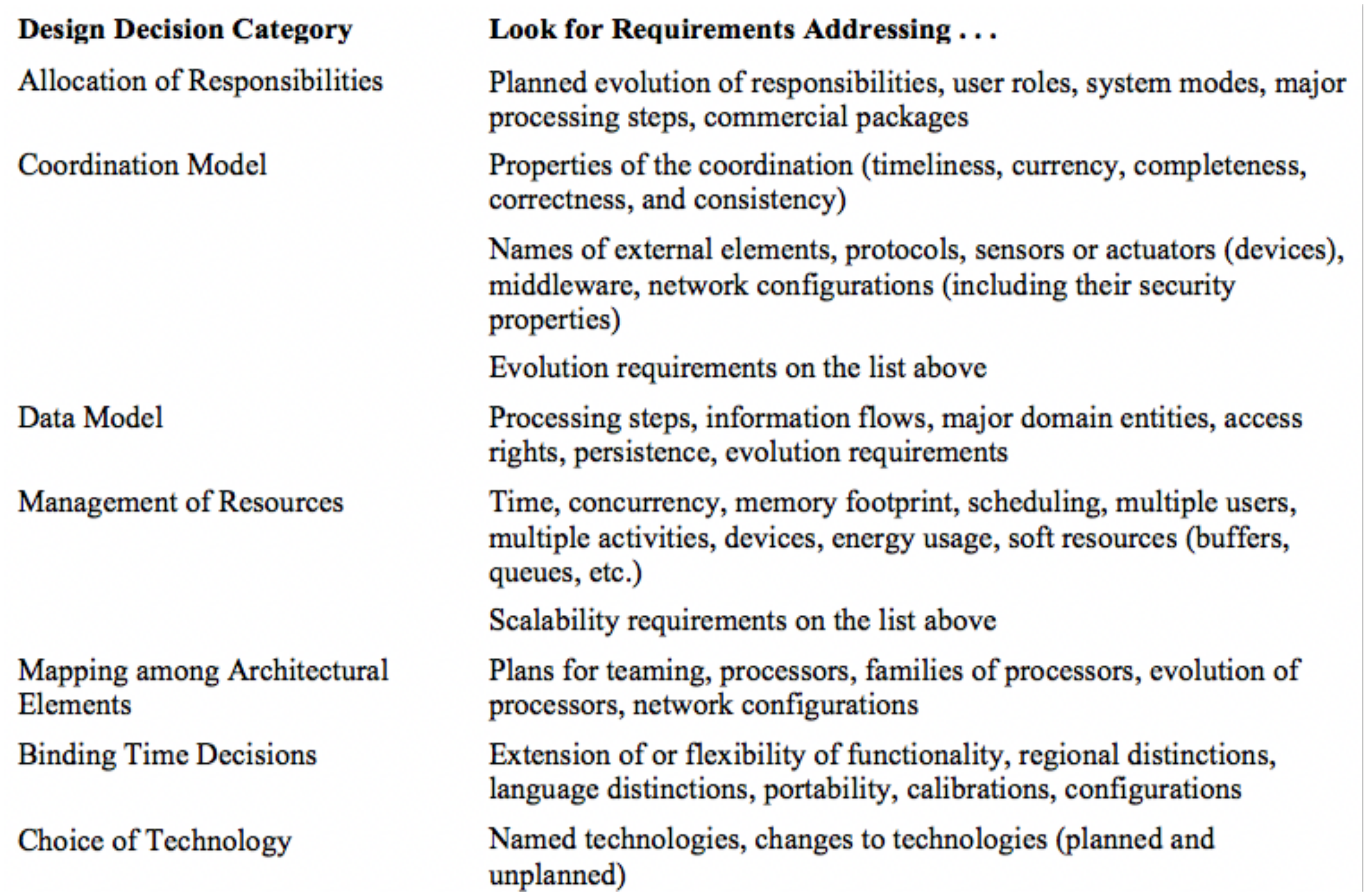
1. It has non-local consequences and
2. The consequences matter to the achievement of the architectural drivers

Decisions of the allocation of the main functionalities establish precedent for how the rest of functionalities should be allocated.

Security Architecture Decisions

- Allocation of responsibilities – assess access needs
- Coordination models – assess secure collaboration
- Data model – ensure security of sensitive data
- Assess the impacts of mapping architecture elements on security
- Resource management – control accesses to monitor and manage the resources
- Binding time – identify binding cases and their security impacts
- Choice of technology – identify the impact of chosen technology on the security

Architecture Drivers – Architecture Decisions



Design Decision Category	Look for Requirements Addressing . . .
Allocation of Responsibilities	Planned evolution of responsibilities, user roles, system modes, major processing steps, commercial packages
Coordination Model	Properties of the coordination (timeliness, currency, completeness, correctness, and consistency)
	Names of external elements, protocols, sensors or actuators (devices), middleware, network configurations (including their security properties)
	Evolution requirements on the list above
Data Model	Processing steps, information flows, major domain entities, access rights, persistence, evolution requirements
Management of Resources	Time, concurrency, memory footprint, scheduling, multiple users, multiple activities, devices, energy usage, soft resources (buffers, queues, etc.)
	Scalability requirements on the list above
Mapping among Architectural Elements	Plans for teaming, processors, families of processors, evolution of processors, network configurations
Binding Time Decisions	Extension of or flexibility of functionality, regional distinctions, language distinctions, portability, calibrations, configurations
Choice of Technology	Named technologies, changes to technologies (planned and unplanned)

Attribute Based Design Steps

1. Review input
2. Establish the iteration goal
3. Choose one or more elements to refine
4. Choose design concepts that satisfy the selected drivers
5. Instantiate architecture elements, allocate responsibilities, and define interface
6. Sketch views and record design decisions
7. Perform analysis of current design and review iteration goal ad achievement of design purpose
8. Iteration if necessary

- Design the system in a set of iterations
- Questions to be answered later
  - How many iterations?
  - Do professionals architect software in iterations?

Use Case 1 – FCAPS system

- The system is an infrastructure that supports Internet communication
- Business goal: Expand the system to support voice-over-IP
- Technical goal: Deploy a set of network servers to support the Network Time Protocol (NTP)
- Goal of the architecture: Produce a sufficient detailed design to support the construction of the system

Use Case 1 – FCAPS system

Main use cases

1. Monitor network status – e.g., identify problematic devices
2. Detect faults
3. Display event history
4. Manage time server – e.g., add and remove a time server
5. Visualize performance data
6. ...

Use Case 1 – FCAPS system

Quality attributes

1. Performance – adding a new server management protocol without changing the core components
2. Availability–management system resumes after fault in 30s
3. Security – records of changes to the systems, i.e., who did what, and when
4. Performance – the system collects performance data within 5 min with no loss
5. ....

Use Case 1 – FCAPS system

Constraints

1. A minimum of 50 simultaneous users could be supported
2. The system must be accessible using browsers
3. An existing relational database must be used
4. 4. .....

Use Case 1 – FCAPS system

Concerns

1. Establish an overall initial system structure
2. Leverage team's knowledge about java technologies
3. Allocate work to members of the development team

Step 1- Review input

- Architecture goal: Produce a sufficient detailed design to support the construction of the system
- Establish primary functional requirements – select the main use cases (e.g., UC1, UC2, and UC7)
- Prioritize quality attributes
- Select constraints
- Select concerns

Step 1- Review input - FCAPS

Set the importance to the customer and the difficulty to implement each of the QAs.

1. Performance – add a new server management protocol without changing the core components (medium, high)
2. Availability–management system resumes after fault in 30 sec (high, medium)
3. Security – record changes to the systems, i.e., who did what, and when? (high, low)
4. Performance – the system collects performance data within 5 min with no losses (high, medium)
5. (There are other QAs)

Step 2 - Establish the Iteration Goal by Selecting Drivers

- Goal: Establish an overall system structure (concern 1)

- Architecture drivers:
  1. All use cases that affect the concern
  2. Quality attributes that affect the concern
  3. Constraints that affect the concern
  4. Concerns related to the concern of the iteration

Step 2 - Establish the Iteration Goal by Selecting Drivers - FCAPS

- Goal: Establish an overall system structure (concern 1)
- Selected architecture drivers
  1. All use cases
  2. QA1- 100% of traps are successfully processed and stored at peak time
  3. QA2- Adding a new server management protocol without changing the core components
  4. QA3- Management system resumes after fault in 30s
  5. QA4- The management system collects all performance data within 5 minutes

Step 2 - Establish the Iteration Goal by Selecting Drivers - FCAPS

- CONS2—The system must be accessible through the web browser
- CONS3– A relational database must be used
- CONS4- The network connection can have low bandwidth
- CRN2- Leverage team's knowledge about Java technologies

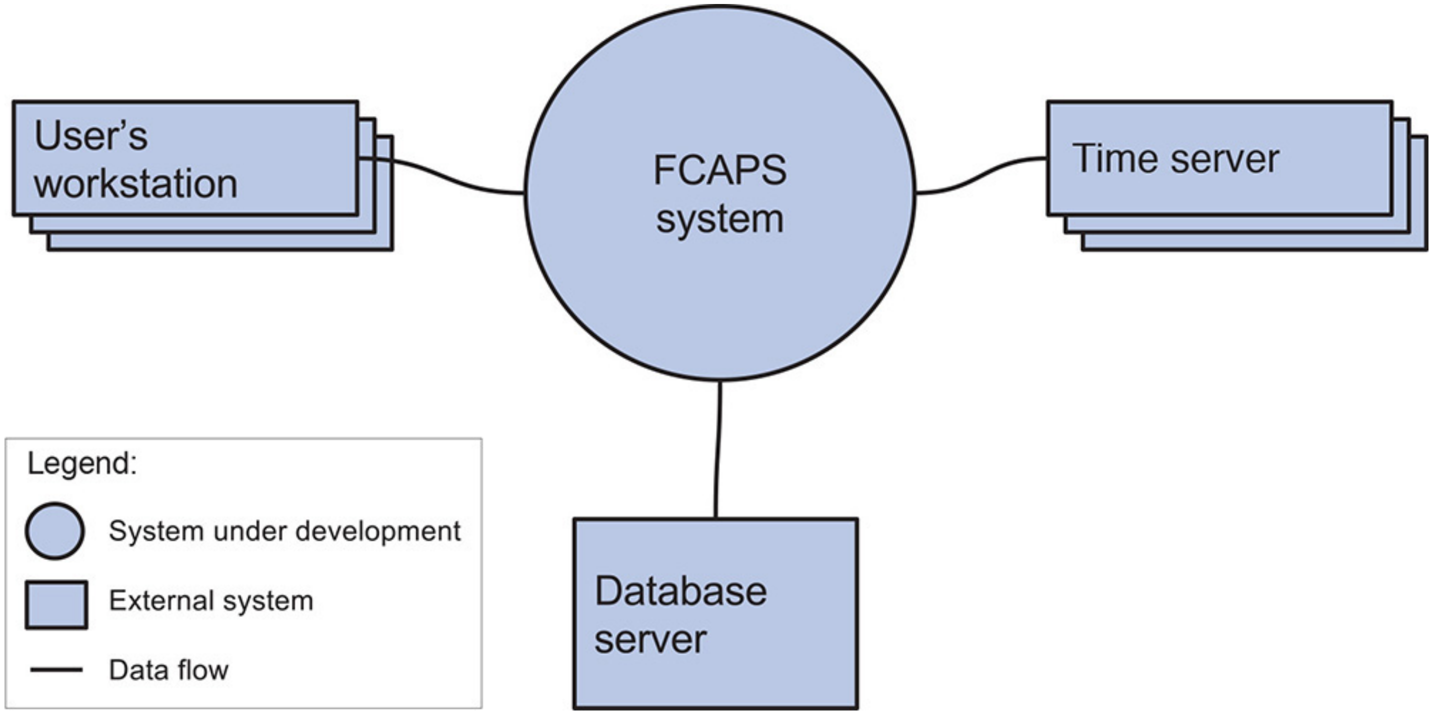
How can we select architecture drivers for an iteration?

Step 3 - Choose One or More Elements to Refine

- Decide on what parts of the system/components should be changed in order to address the selected architecture drivers.

Step 3 - Choose One or More Elements to Refine - FCAPS

- What parts of the system do we need to change?



Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

- We select a set of architecture drivers that can be addressed using one or a set of architecture concepts (i.e., reference architecture, patterns and tactics)

Selection techniques

1. Pro+/cons
2. Cost benefit analysis
3. Strengths, weaknesses, opportunities and threats

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers - FCAPS

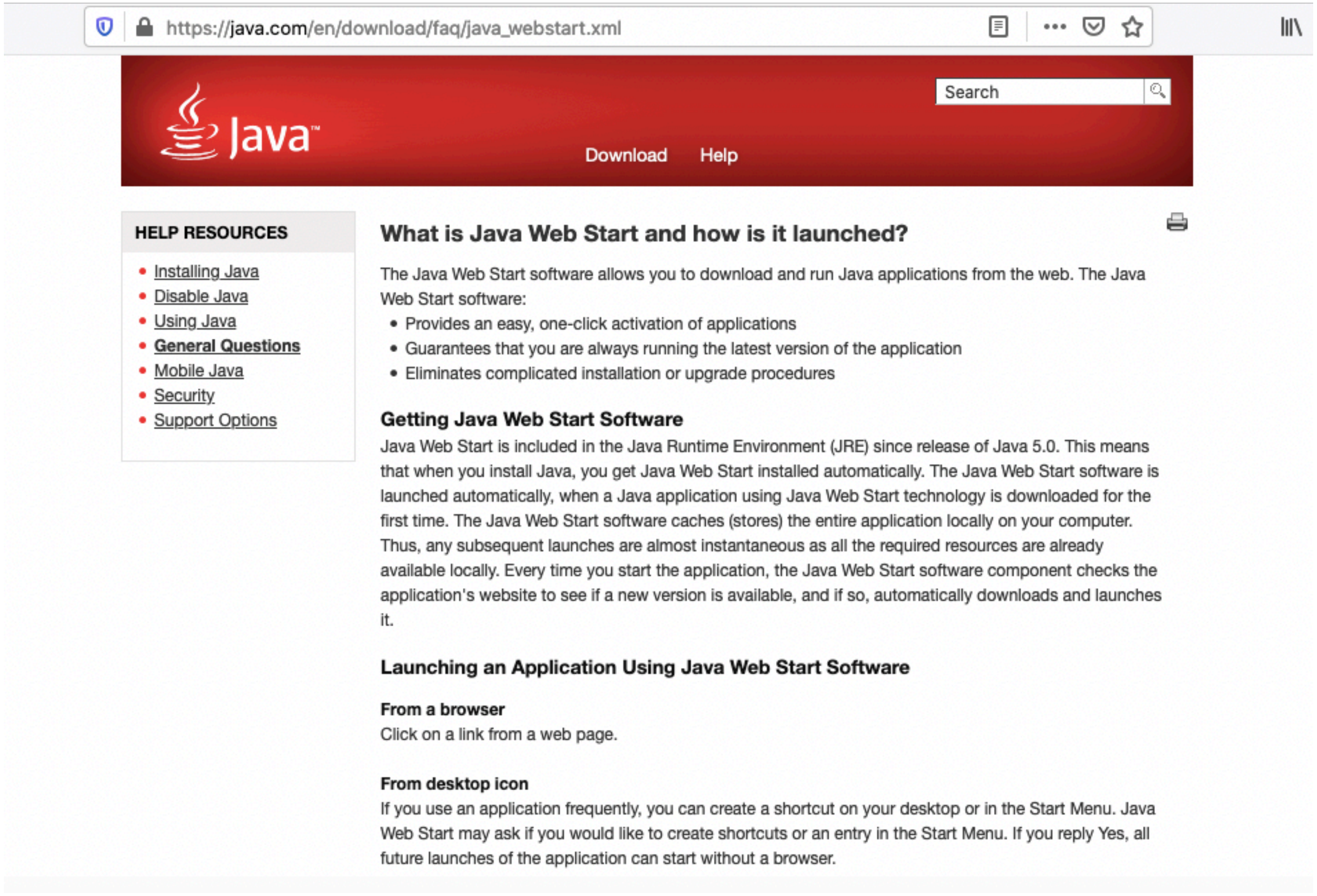
Which option addresses the following drivers the best?

1. UC1- Visualize performance data
2. CONS 2- The system must be accessed using browser

Options:

1. Web application
2. Rich client application
3. Rich internet application
4. Mobile application

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers - FCAPS



HELP RESOURCES

- Installing Java
- Disable Java
- Using Java
- **General Questions**
- Mobile Java
- Security
- Support Options

What is Java Web Start and how is it launched?

The Java Web Start software allows you to download and run Java applications from the web. The Java Web Start software:

- Provides an easy, one-click activation of applications
- Guarantees that you are always running the latest version of the application
- Eliminates complicated installation or upgrade procedures

Getting Java Web Start Software

Java Web Start is included in the Java Runtime Environment (JRE) since release of Java 5.0. This means that when you install Java, you get Java Web Start installed automatically. The Java Web Start software is launched automatically, when a Java application using Java Web Start technology is downloaded for the first time. The Java Web Start software caches (stores) the entire application locally on your computer. Thus, any subsequent launches are almost instantaneous as all the required resources are already available locally. Every time you start the application, the Java Web Start software component checks the application's website to see if a new version is available, and if so, automatically downloads and launches it.

Launching an Application Using Java Web Start Software

From a browser
Click on a link from a web page.

From desktop icon
If you use an application frequently, you can create a shortcut on your desktop or in the Start Menu. Java Web Start may ask if you would like to create shortcuts or an entry in the Start Menu. If you reply Yes, all future launches of the application can start without a browser.

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers - FCAPS

Which option addresses the following drivers better?

1. UC 5- Visualize performance data
2. CONS 2- The system must be accessed using browser
3. CONS 3 - Use an existing database

Options:

1. N-tier deployment pattern
2. Three-tier deployment pattern
3. 2-tier deployment pattern

How can we address the following requirement?

- QA 2 - Management system resumes after fault in 30 seconds

Option(s):

1. Availability: use replication

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers - FCAPS

Which option addresses the following driver the best?

- CRN2- Leverage team’s knowledge about Java technologies

Options:

- Eclipse Standard Widget Toolkit
- Swing Java framework

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers - FCAPS

- How can we know the possible options?
- How can we identify the best option?
- Appendix A of the textbook provides a catalogue of design concepts used in the 3 case studies discussed in the book.

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface

1. Customize the reference architecture
   - e.g., add and remove components to the reference architecture
2. Adapt the pattern to the problem
   - e.g., define the layers
3. Use design concept to instantiate the tactic
   - e.g., use a security pattern for authentication
4. Use externally developed component
   - e.g., select framework

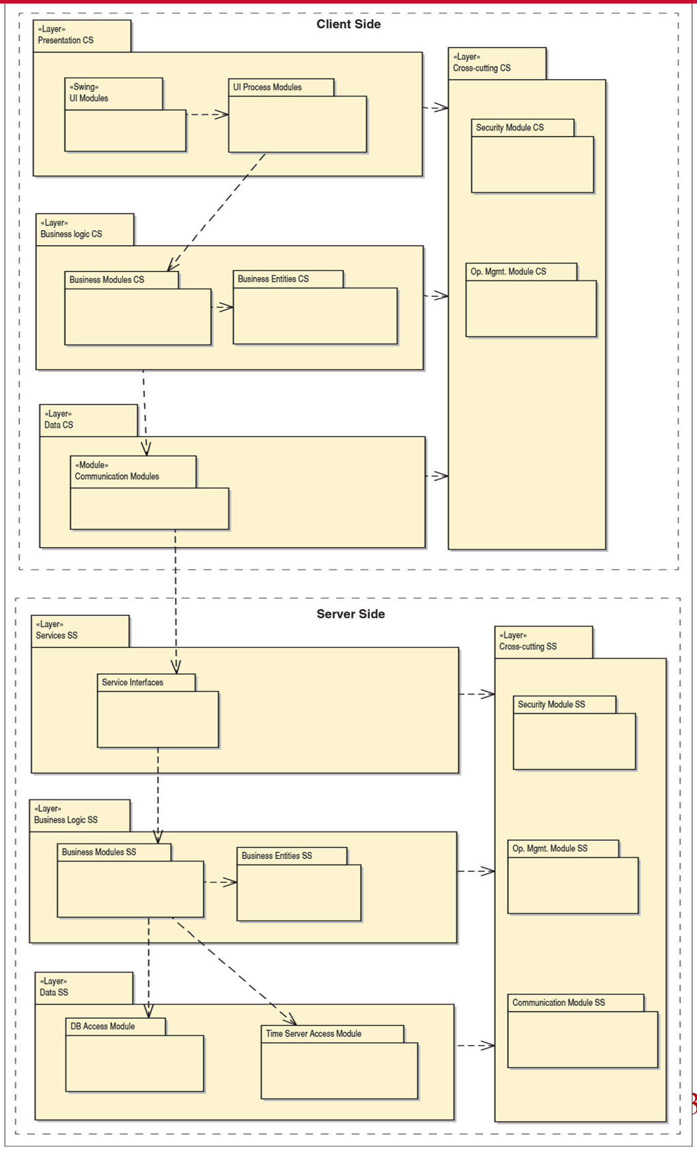
Step 6 - Sketch Views and Record Design Decisions

- Instantiate the decisions by creating architecture elements and relationships among these elements
  - Components diagram
  - Communication diagram
  - Concurrency model
  - • etc.

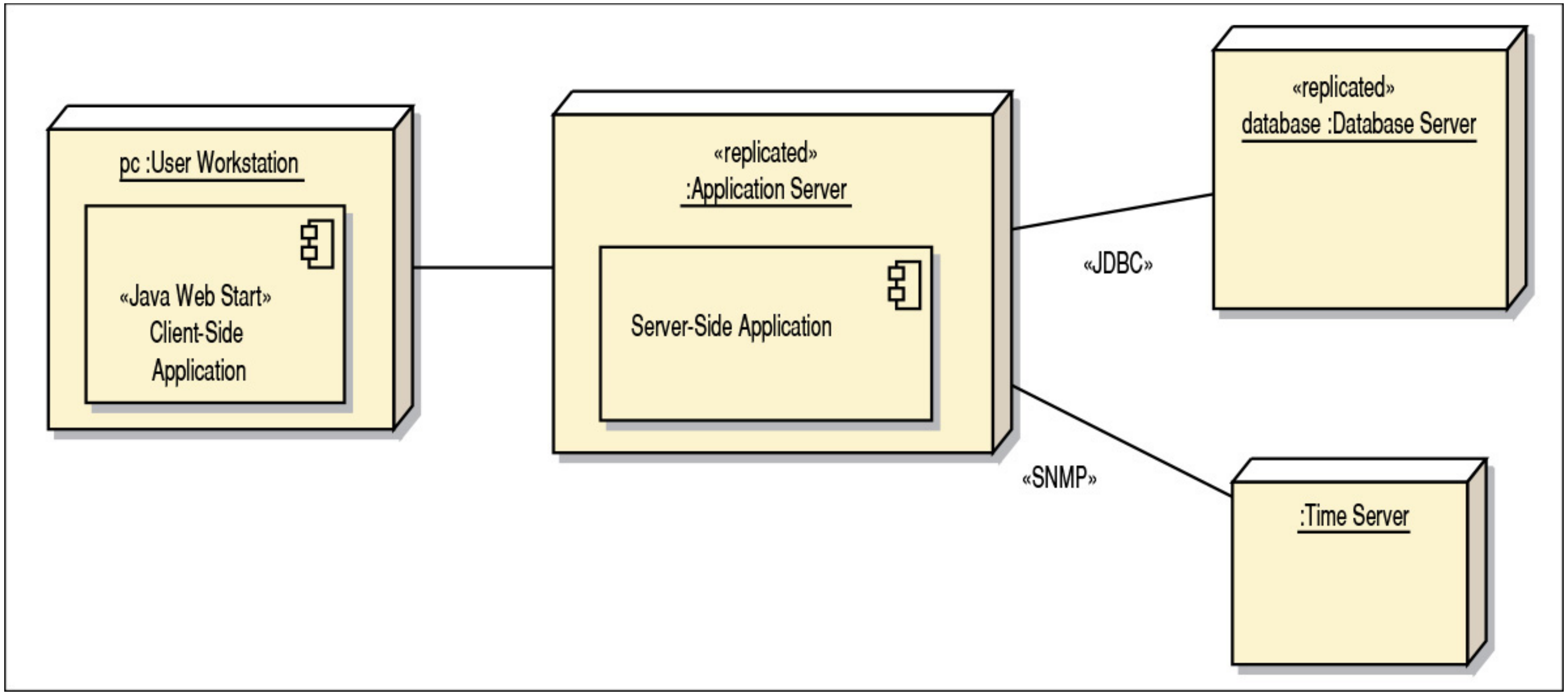
- Define the interfaces of the components

Step 6 - Sketch Views and Record Design Decisions - FCAPS

Why did the architect use the three-layers pattern?



Step 6 - Sketch Views and Record Design Decisions - FCAPS



UML deployment diagram – Use of the three-tier architecture

Step 7 - Perform Analysis of Current Design - FCAPS

- Assess how good are the solutions for all the architecture drivers.
- Identify potential conflicts between selected architecture concepts
  - E.g., single-point access control and components replication

Step 7 - Perform Analysis of Current Design - FCAPS

Completely addressed

1. CONS 2- The system must be accessed through browsers

Not addressed architecture drivers

1. QA1- 100% of traps are successfully processed and stored at peak time
2. QA4- The management system collects all performance data within 5 minutes
3. CON5-Time servers discard data older than 5 min
4. CON6-Events from the last 30 days must be stored
5. CRN3-Allocate work to members of the development team

Step 8 - Iteration If Necessary - FCAPS

- The current sketches do not address all the architecture drivers
- The sketches are not detailed enough

=> Make iteration 2

Step 2 - Establish the Iteration Goal by Selecting Drivers – Iteration 2

- Goal: Define an architecture structure to support the primary functionalities

- UC1- Monitor network status
- UC2- detect fault
- UC3- display event history

- CRN- Allocate work to the team members

Step 3 - Choose One or More Elements to Refine - FCAPS

- Recall that we have:
  - Client and server sides
  - Layers at the client side and server side
- We need to refine these components to support the three selected use cases.

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

- Develop the domain model of the system
  - A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest
- Associate the use cases to the domain model to make sure all use cases are covered
- Decompose the objects into the layers and distribute the resulting components into these layers
- Support enterprise application development
  - Option: Use Spring framework and Hibernate

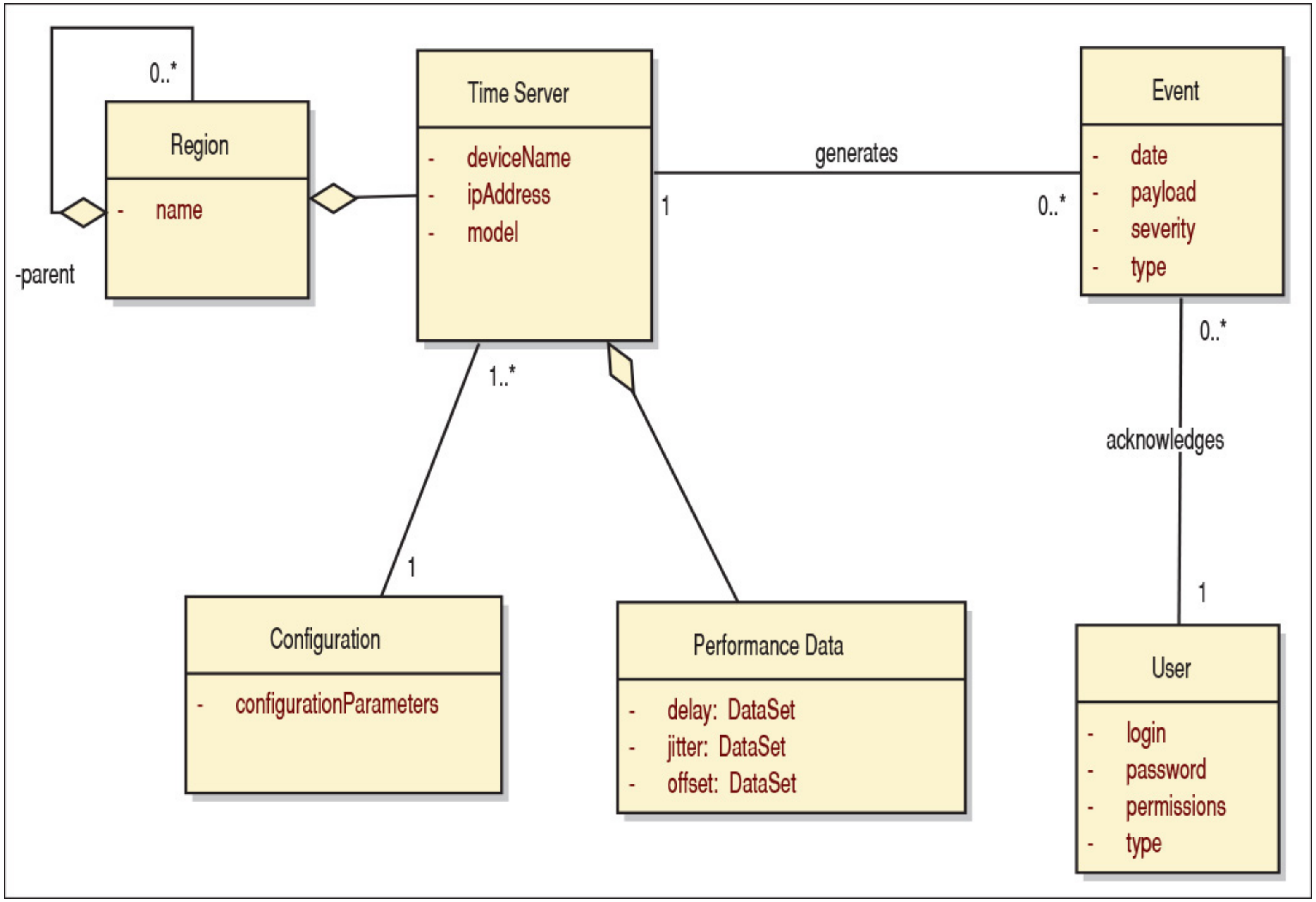
    Widely used frameworks

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface --FCAPS

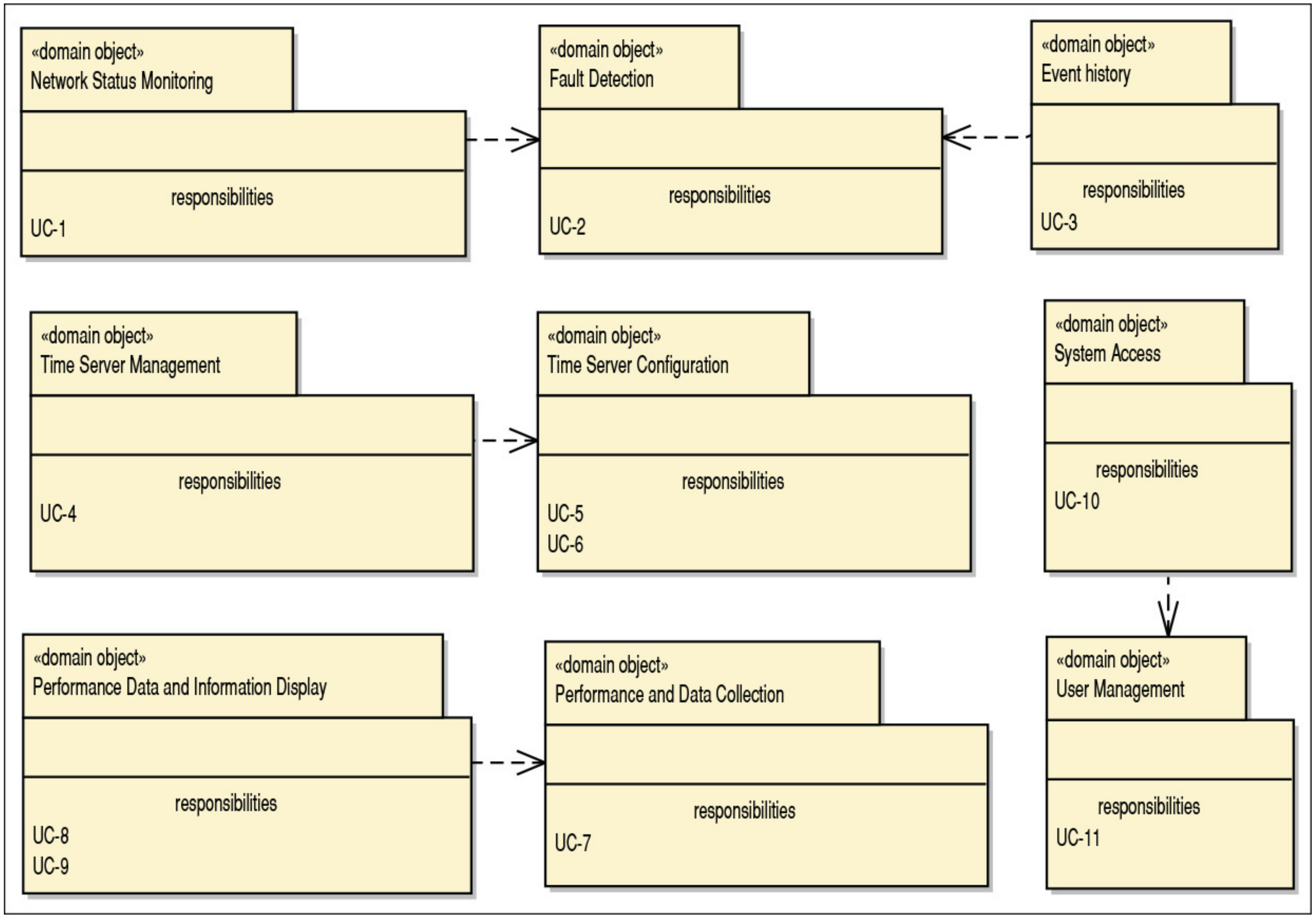
- The framework uses an inversion control approach
  => Associate software components to the framework modules

- Associate the modules to the data layer

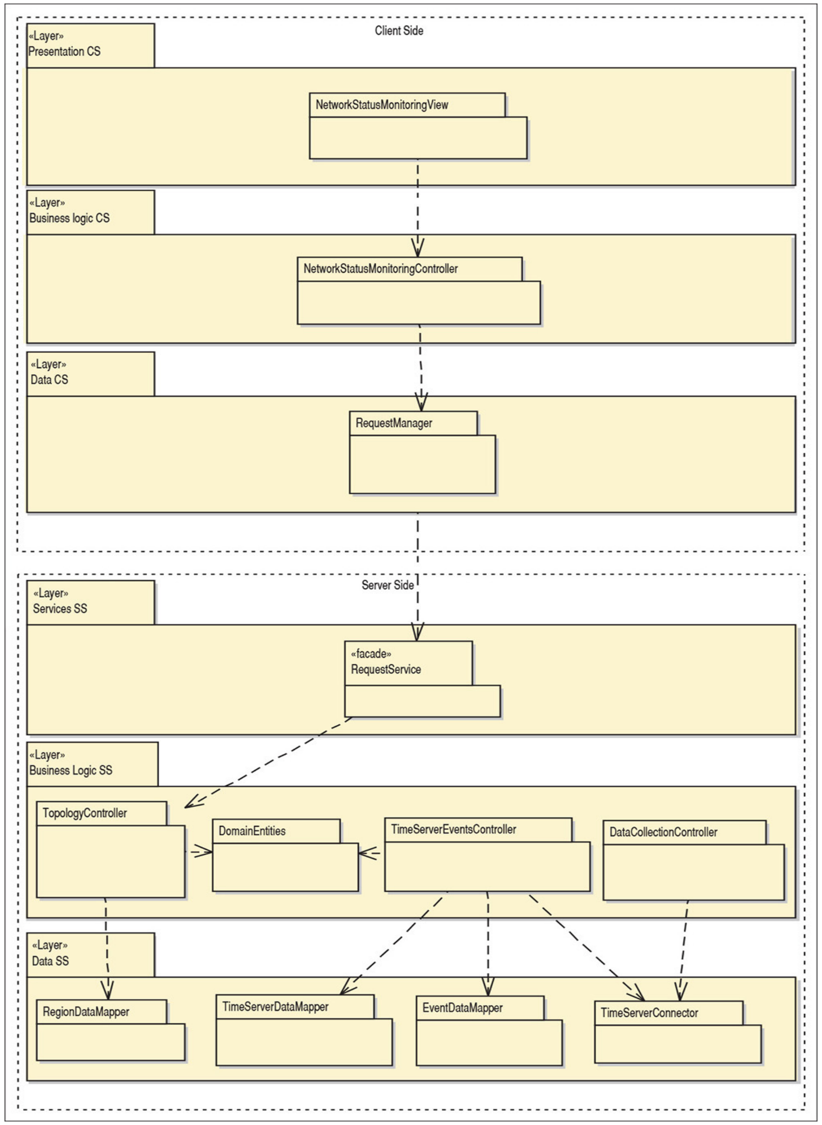
Step 6 - Sketch Views and Record Design Decisions - FCAPS



Step 6 - Sketch Views and record Design Decisions - FCAPS

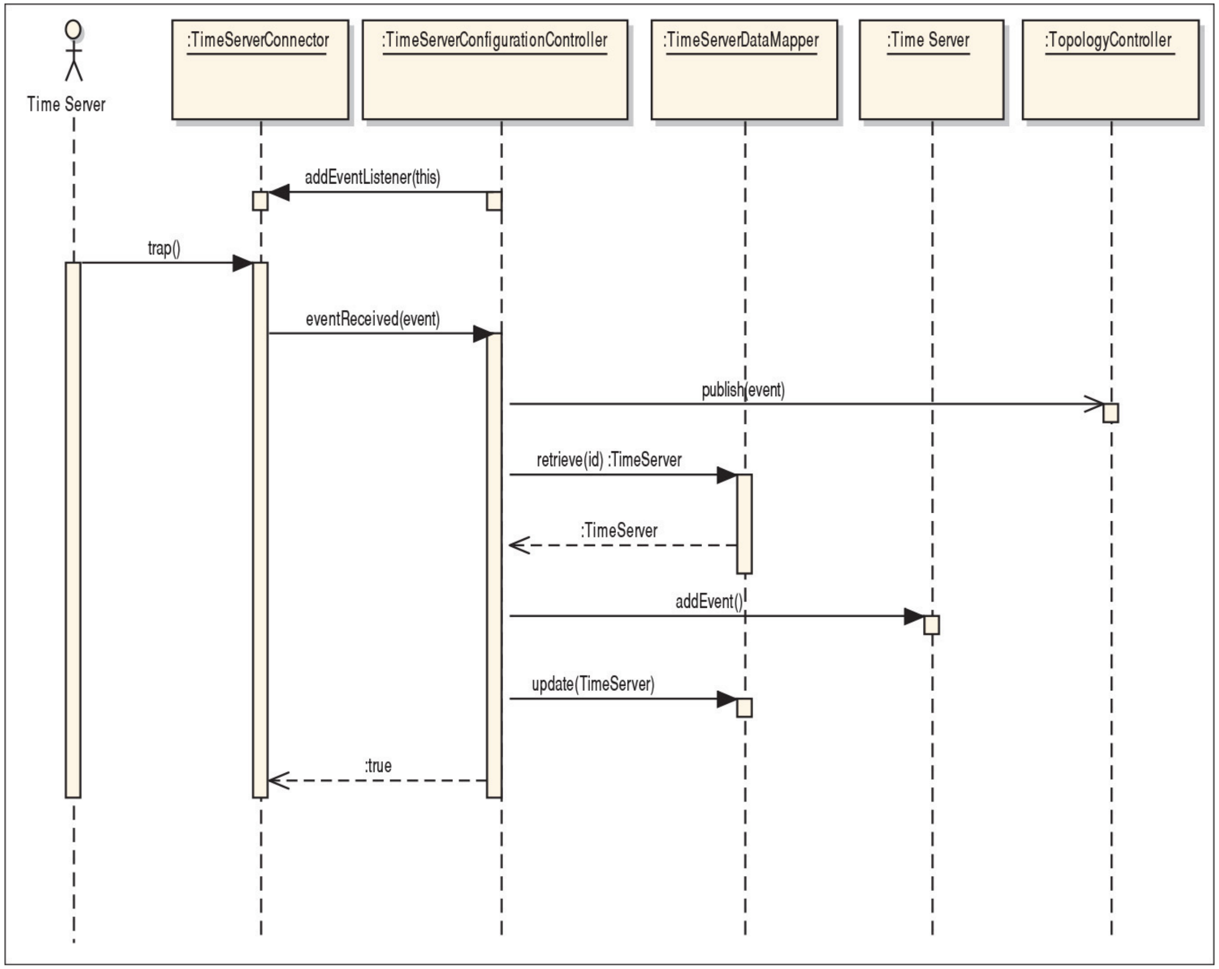


Step 6 - Sketch Views and Record Design Decisions - FCAPS



Step 6 - Sketch Views and record Design Decisions - FCAPS

Sequence diagram for use case detect fault



Step 7 - Perform Analysis of Current Design - FCAPS

- Completely addressed
  - All use cases are addressed
- Partially addressed
  - All quality attributes
- CRN 3—work assignment matrix is created

Step 8 - Iteration If Necessary - FCAPS

- The current sketches do not address: QA3- Management system resumes after application fault in 30 seconds

Step 2 - Establish the Iteration Goal by Selecting Drivers – Iteration 3

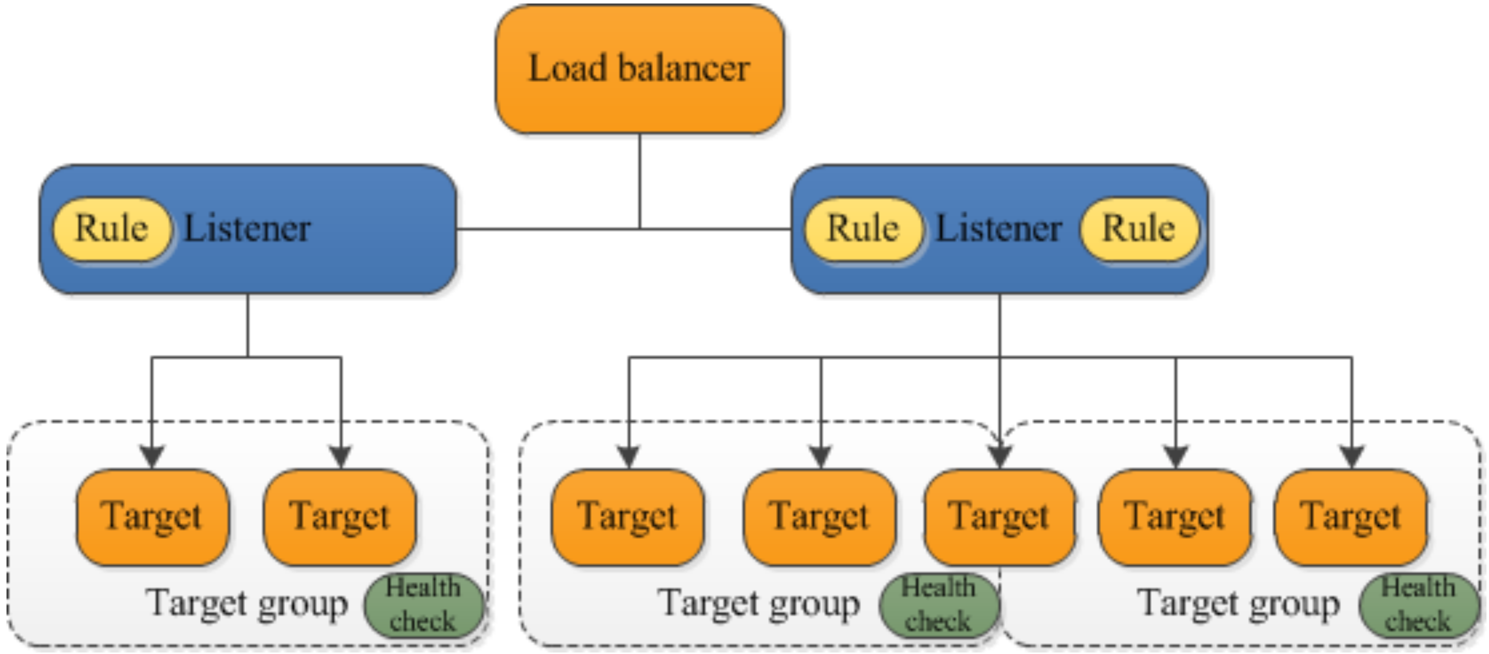
- Goal: Address QA3- Management system resumes in 30 seconds after application fault

Step 3 - Choose One or More Elements to Refine - FCAPS

1. Application server
2. Database server

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

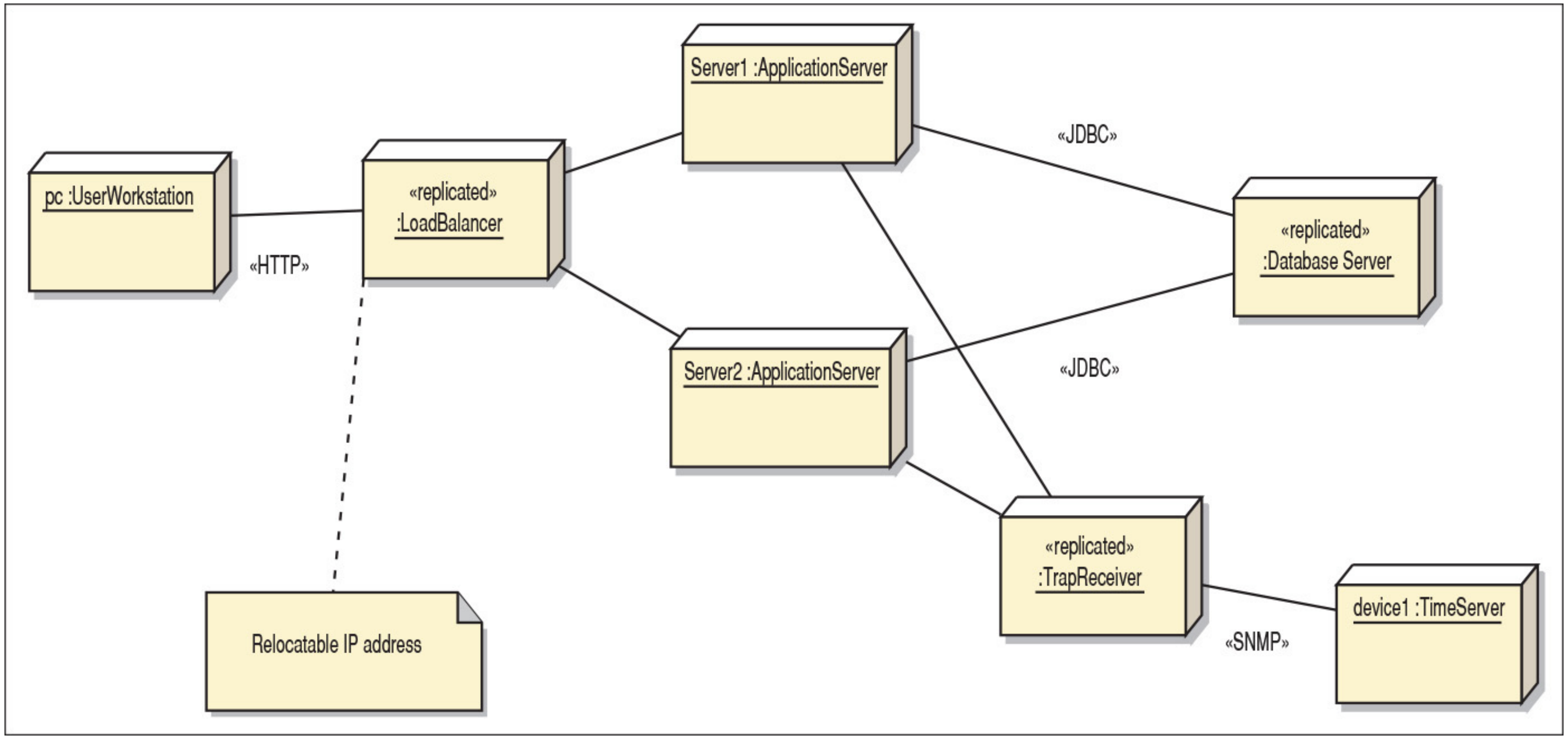
- Use of active redundancy tactic: replicating the application server
- Use of messages queue to ensure no messages are lost



Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface -- FCAPS

- Deploy messages queue on a separate node. The node is replicated but only one node services the time servers
- Use load balancer to distribute the load between the application servers. The load balancer excludes automatically failed nodes

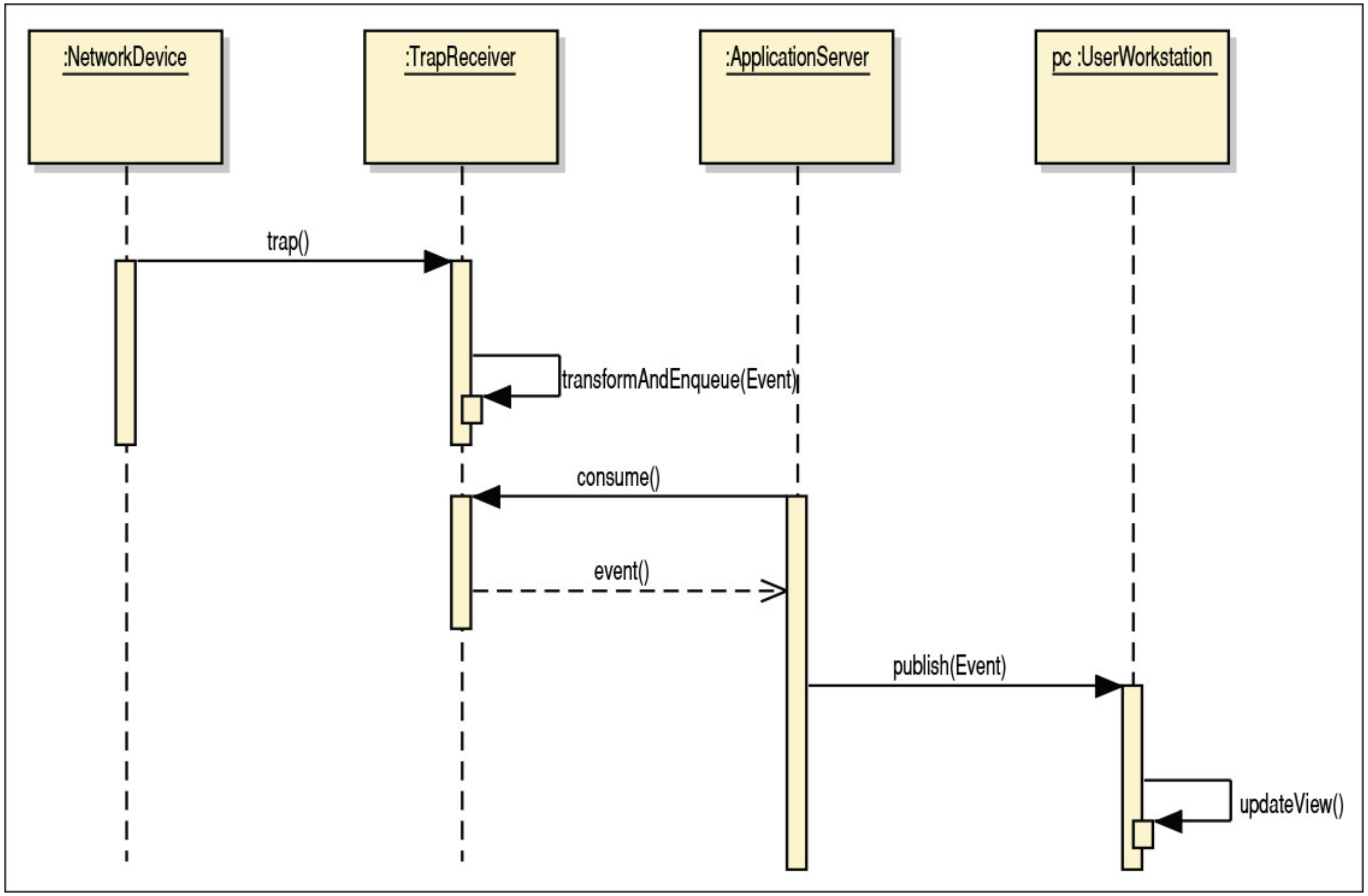
Step 6 - Sketch Views and record Design Decisions - FCAPS


pc :UserWorkstation
«HTTP»
«replicated» :LoadBalancer
Server1 :ApplicationServer
Server2 :ApplicationServer
«JDBC»
«JDBC»
«replicated» :Database Server
«replicated» :TrapReceiver
«SNMP»
device1 :TimeServer
Relocatable IP address

UML deployment diagram of the system

Step 6 - Sketch Views and Record Design Decisions - FCAPS

New sequence diagram for use case detect fault



Step 7 - Perform Analysis of Current Design - FCAPS

- QA1- 100% of traps are successfully processed and stored at peak time – separate replicated queue management node
- QA2- Add new management protocol with no major changes to the main system components – no decision
- QA3- Management system resumes operations in 30 sec – use of the load balancer and replication of the application server
- QA4- Management system collects all performance data within 5 minutes – there are enough application servers

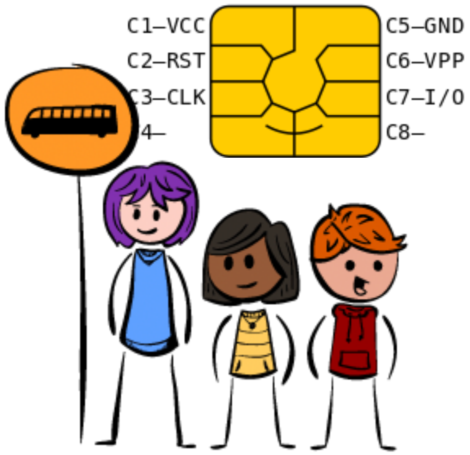
Step 7 - Perform Analysis of Current Design - FCAPS

- QA5- A user displays the list of events history of the last 24h in 1 second – application server is replicated
- QA6-Have trace of updates to the system – It becomes a use case
- CONS1- A minimum of simultaneous 50 users must be supported – application server is replicated
- CONS2—The system must be accessible through the web browser—supported
- CONS3– A relational database must be used-no decision

Step 7 - Perform Analysis of Current Design - FCAPS

- CONS4- The network connection can have low bandwidth--no decision
- CON5-Time servers discard data older than 5 min—no decision
- CON6-Events from the last 30 days must be stored—no decision
- CRN2- Leverage team's knowledge about Java technologies—use spring framework
- CRN3—allocate work to members of the development team

Exercise – Pay With Travel Card



Exercise – Pay With Travel Card

Design an architecture for a system to “buy bus ticket.” Architecture drivers are:

1. Passengers can charge their cards,
2. Passengers scan their cards when they get-in,
3. Passengers scan their cards when they get-out,
4. The in-bus device computes the trip price, deducts the price from the balance in the card and registers the trip in the card and in the remote service,

Exercise – Pay With Travel Card

Other architecture drivers

1. Customers should get-in in less than one minute
2. The bus may travel in areas where there is no signal.
3. Consider the specialization of the developers
4. The system should have no single point of failure
5. The company expects to extend the system with data analytics capabilities

Comments on Participation 14 Submissions

- Most of the students developed an object model for the project, which a good start – The models do not however cover all use cases.
- Some students provided architecture sketches that address some QAs
- Working on iteration allows for systematic thinking.

Attribute Based Design Steps

1. Review input
2. Establish the iteration goal
3. Choose one or more elements to refine
4. Choose design concepts that satisfy the selected drivers
5. Instantiate architecture elements, allocate responsibilities, and define interface
6. Sketch views and record design decisions
7. Perform analysis of current design and review iteration goal and achievement of design purpose
8. Iteration if necessary

Review Input - Pay With Travel Card

Design goal: Prepare an architecture for the city of Ames’ call for proposal

Primary functional requirements:

- Passengers can charge their cards
- Passengers scan their cards when they get-in
- Passengers scan their cards when they get-out,
- The in-bus device computes the trip price, deducts the price from the balance in the card and registers the trip in the card and in the remote service,

Review Input - Pay With Travel Card

Quality attributes scenarios

QA1 - Customers should get-in in less than one minutes. (high, high)

QA2 - The buses may travel in areas where there is no signal. (high, low)

QA3 - Consider the specialization of the developers. (medium, medium)

QA4 - The system should have no single point of failure. (high, high)

QA5 - The company expects to extend the system with data analytics capabilities. (low, low)

Iteration 1—Define the Overall Structure of the System and Support Main Use Cases

- Iteration goal: Define the overall structure of the system and support main use cases

- Selected architecture drivers
  - Select all use cases

Step 3 - Choose One or More Elements to Refine

- Refine all the system

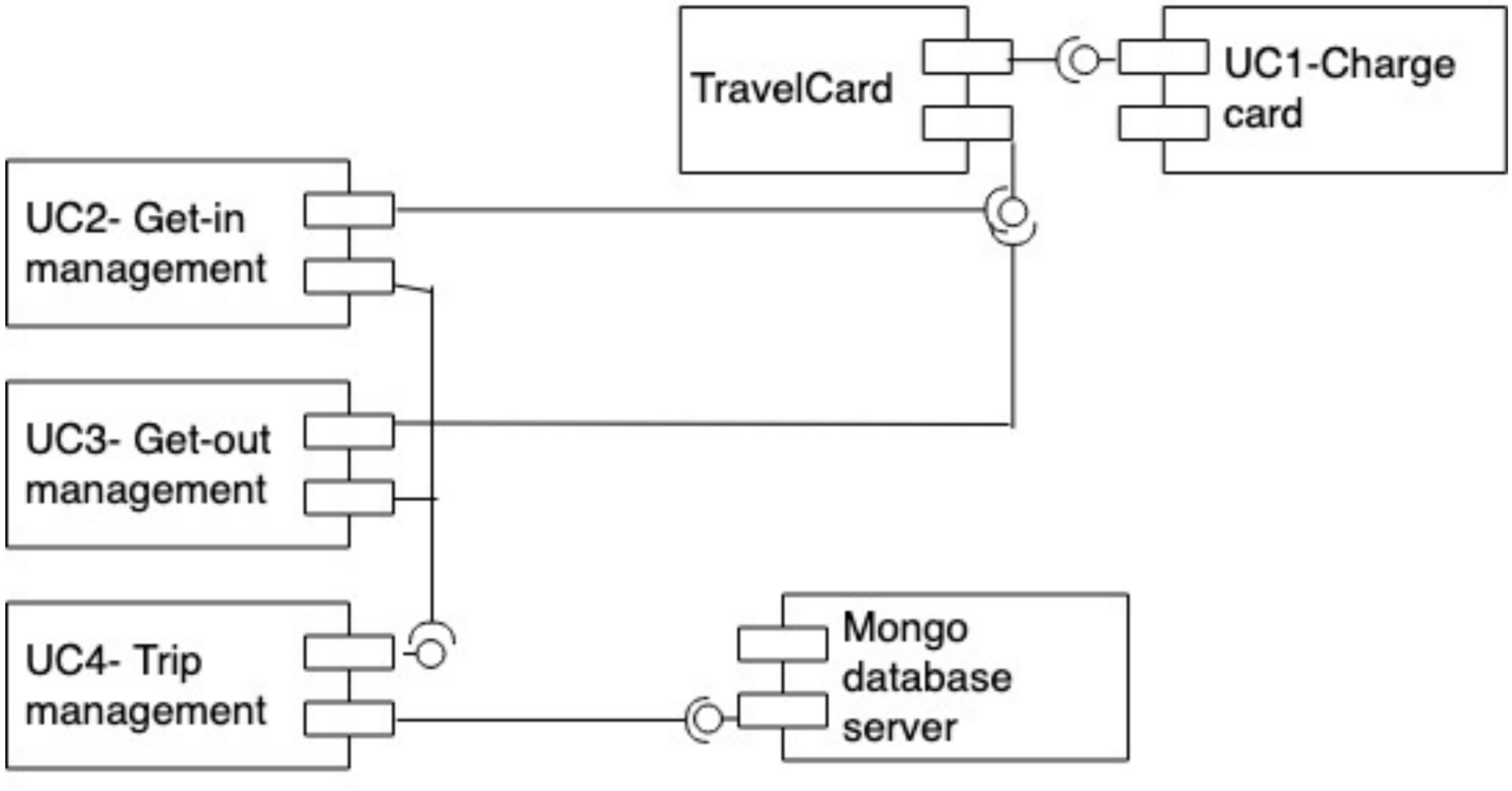
Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

- Selection of architecture reference
  - Microservices
  - Client/Server
  - Desktop
  - Mobile application
- Associate the use cases to the domain model to make sure that all use cases are covered

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface

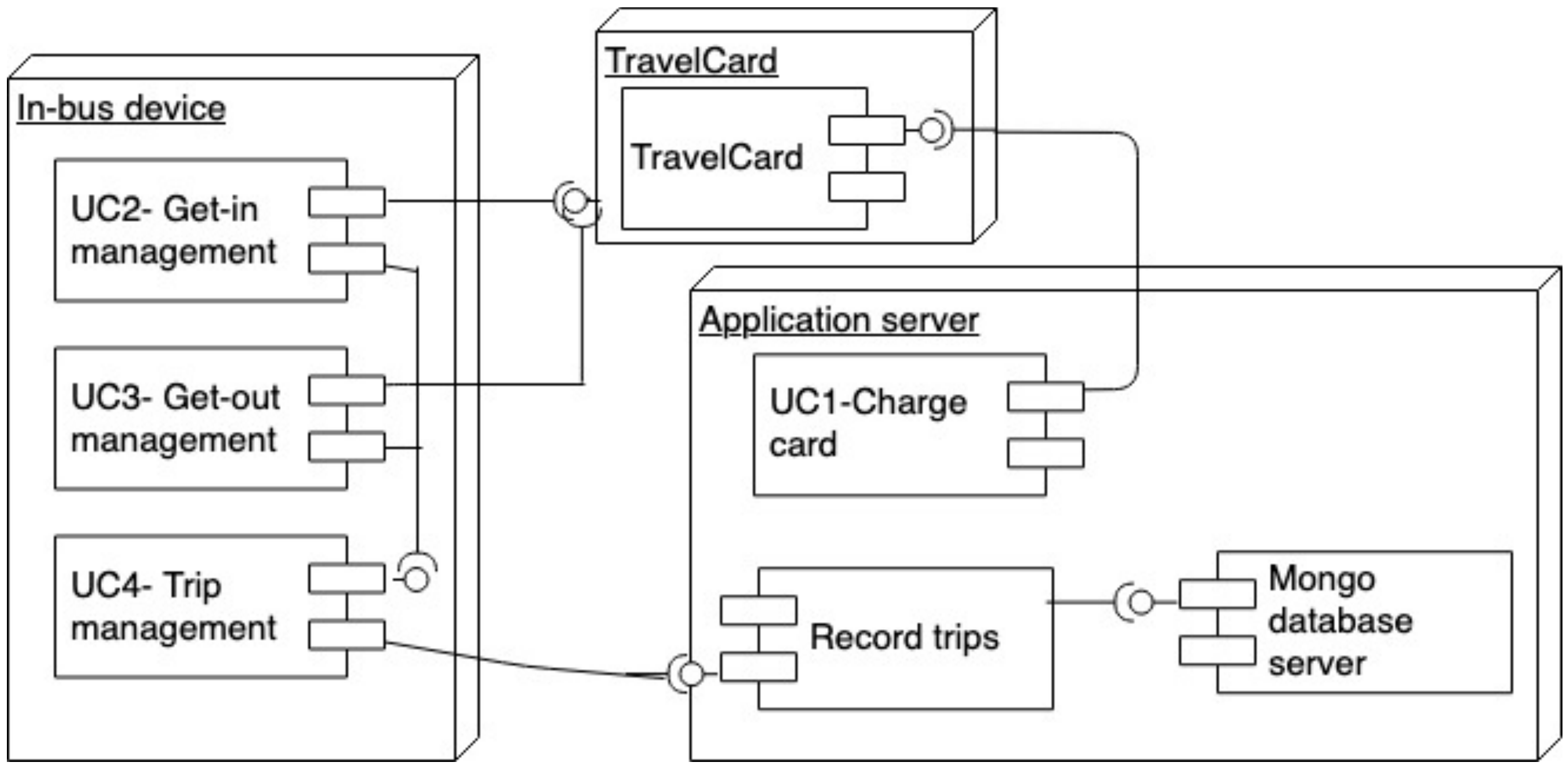
- Use Node.JS for the application server
- Select NXP MIFARE DESFIRE
- Use Java to program the smart card

Step 6 - Sketch Views and Record Design Decisions



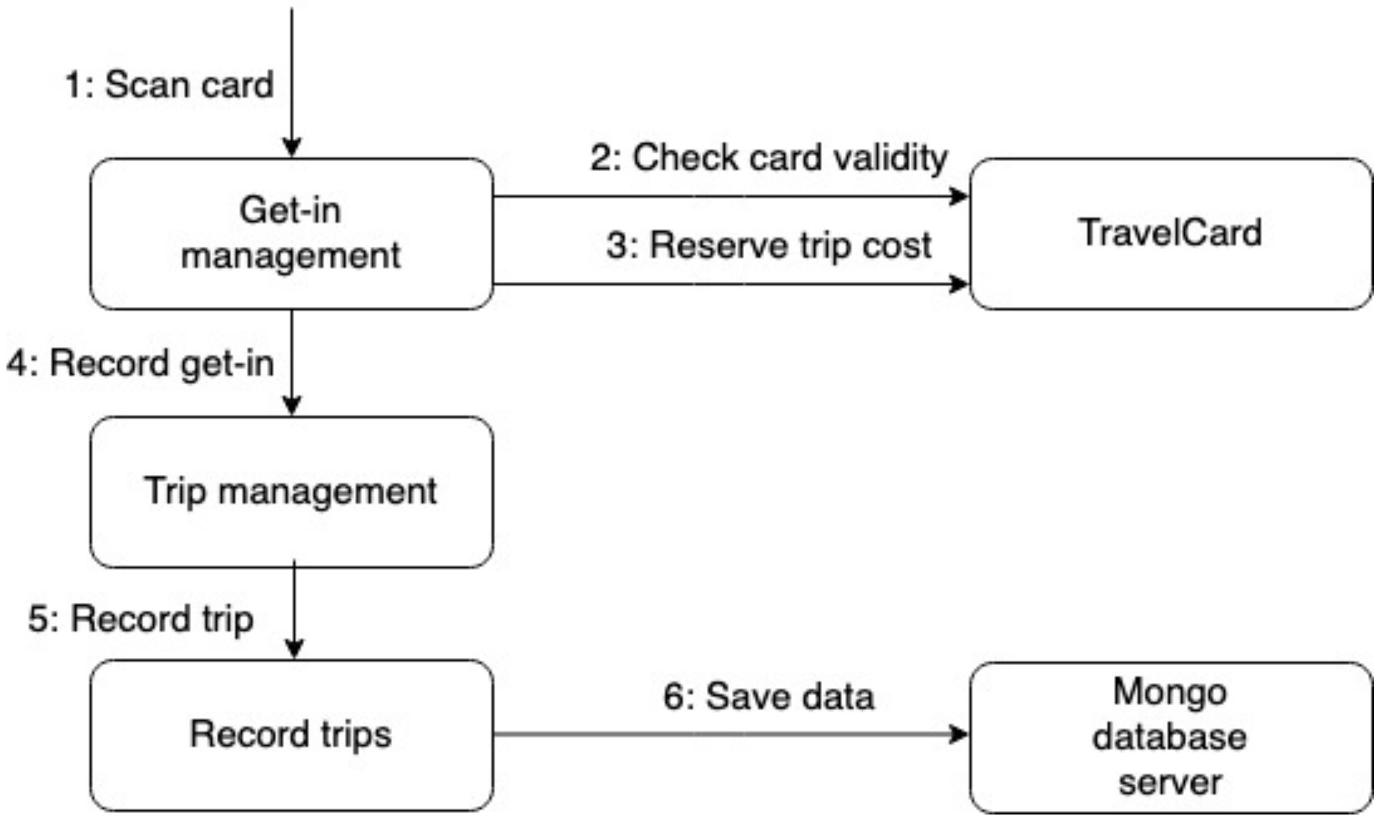
- UML component diagram for the system

Step 6 - Sketch Views and Record Design Decisions



- UML deployment diagram for the system

Step 6 - Sketch Views and Record Design Decisions



UML communication diagram for the get-in use case

Step 7 - Perform Analysis of Current Design

✓

Completely addressed
All use cases

Partially addressed
None

⚠

No addressed
All quality attributes

Step 8 - Iteration If Necessary

The current sketches do not address the quality attributes

QA1 - Customers should get-in in less than one minutes.

QA2 - The buses may travel in areas where there is no signal.

QA3 - Consider the specialization of the developers.

QA4 - The system should have no single point of failure.

QA5 - The company expects to extend the system with data analytics capabilities.

Iteration 2: Address a Set of Quality Attributes

Goal: Address a selected set of quality attributes

Selected quality attributes

- QA1 - Customers should get-in in less than one minutes
- QA4-The system should have no single point of failure

Step 3 - Choose One or More Elements to Refine

To refine all the system

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

Selected quality attributes

- QA1 - Customers should get-in in less than one minute
- QA4-The system should have no single point of failure

Scenario:

- 100 buses
- Waiting line could reach 30 persons. (
- The system processes 3 requests / seconds.
- 2 get-in devices/bus, each takes 1 seconds to process a request

Options

- Use Queue management pattern
- Use master/slave pattern to implement redundancy

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface

Scenario:

- 100 buses
- Waiting line could reach 30 persons
- The bus stops every 10 min
- 2 get-in devices/bus, each takes 1 seconds to process a request

Options

- Use Queue management pattern
- Use master/slave pattern to implement redundancy
  - A server processes 3 requests per second
  - How many servers do we needed?
  - How much memory do we need for the queue?

(Network bandwidth is sufficient.)

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface

- (Arrival rate) λ = 5; (Processing rate) μ = 3
- Utilization: ρ = λ/(cμ) = 5/3 *X* 1 = 1.66
- Utilization: ρ = 5/3 *X* 2 = 0.833
- Number in the Queue $L_q = \frac{\rho^2}{1-\rho} = 4.3$
- Wait in the Queue Wq = L_q /λ= 0.86 s.
- Wait in the System W=Wq + 1/μ= 1.19s.
- Number in the System L=λW =5.96.
- Proportion of time the server is idle = 1 – ρ = 0.167.

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface

Scenario:

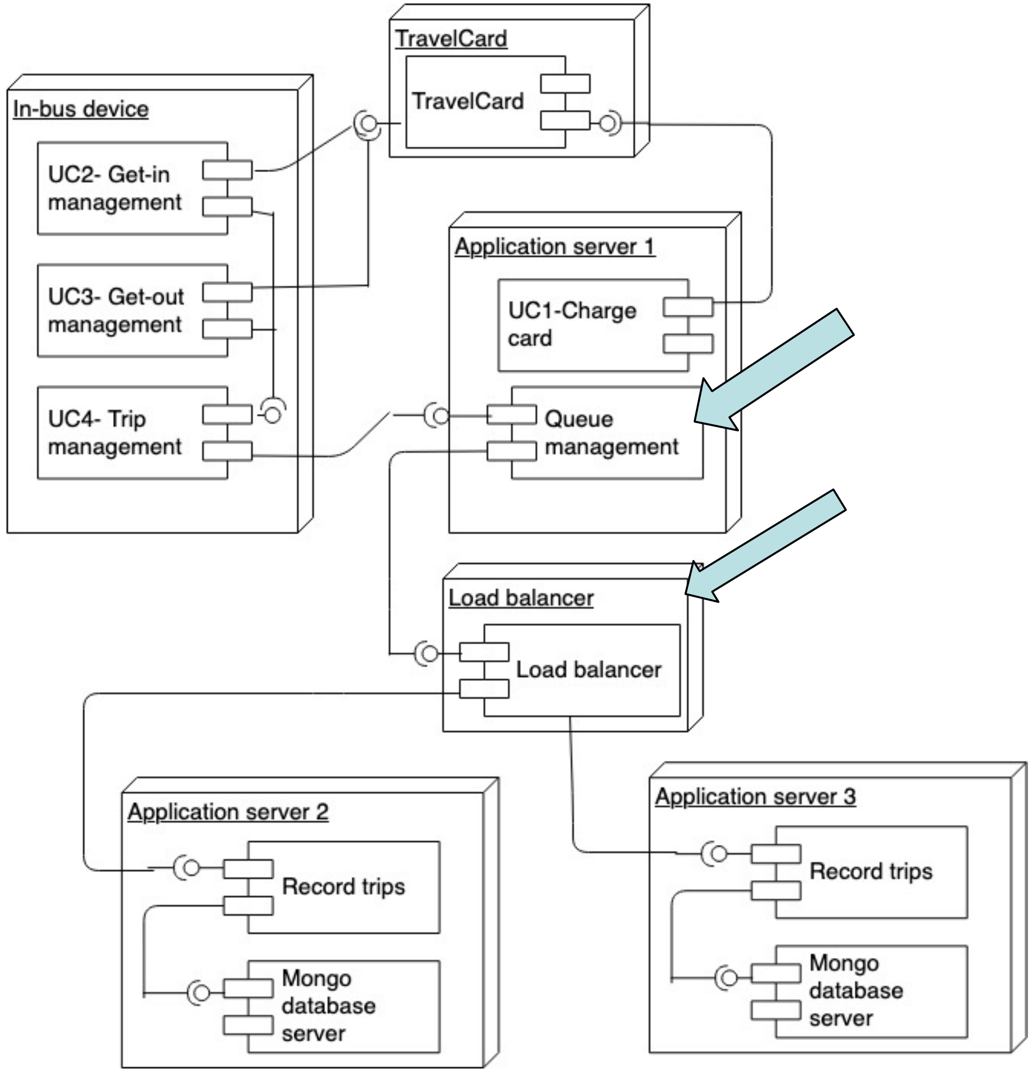
- 100 buses
- Waiting line could reach 30 persons
- The system processes 3 requests /sec.
- 2 get-in devices/bus, each takes 1 seconds to process a request

Options

- Use Queue management pattern – RabbitMQ
- Use master/slave pattern to implement redundancy – hardware load balancer
- Other alternatives include the use of AWS services

Step 6 - Sketch Views and Record Design Decisions

- UML deployment
- diagram for the system



Step 7 - Perform Analysis of Current Design

Completely addressed

- All use cases
- QA1
- QA4

Partially addressed

No addressed

- QA2
- QA3
- QA5

Step 8 - Iteration If Necessary

The current sketches do not address the quality attributes

- QA2 - The buses may travel in areas where there is no signal.
- QA3 - Consider the specialization of the developers
- QA5 - The company expects to extend the system with data analytics capabilities

Iteration 3: Address a Set of Quality Attributes

Goal: Address a selected set of quality attributes

Selected quality attribute:

- QA2 - The buses may travel in areas where there is no signal.

Step 3 - Choose One or More Elements to Refine

Refine the components at the bus device and components at the applications servers

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

Selected quality attributes

- QA2 - The buses may travel in areas where there is no network signal.

Scenario:

- 100 buses
- Waiting line could reach 30 persons
- The system processes 3 requests per second.
- 2 get-in devices/bus, each takes 1 seconds to process a request

Options

- Use buffering pattern
- Reserve fixed trip cost , e.g., 5$

Step 4 - Choose Design Concepts that Satisfy the Selected Drivers

Analysis

- Buffering is used to address QA2
- QA1 – Cannot be achieved in areas where there is no connection

➔ Should we cancel the use of queue management pattern and master/slave pattern?

- The solution addresses QA4 (system should have no single point of failure)
- The average rate of the requests coming to the servers may become higher than the average of rate of requests processed by the servers ➔ Keep the solution

Step 5 - Instantiate Architecture Elements, Allocate Responsibilities, and Define Interface

Scenario:

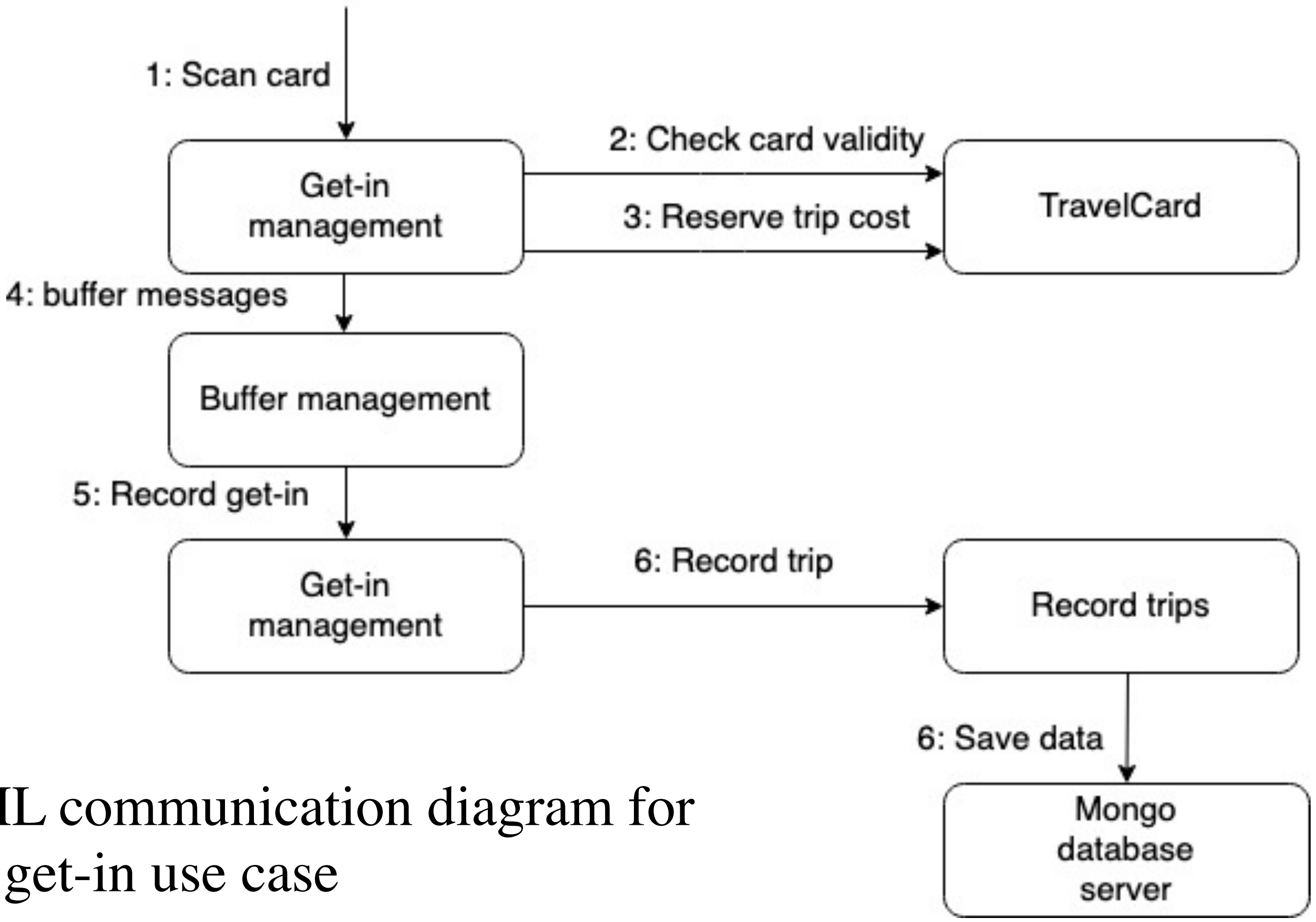
- Connection could be lost for 15 min
- The bus can make up to 2 stops in 15 min
- A bus can take up to 30 passengers in a stop.

Options

- Buffer size: 10*2*30 records
- This requires: 3.34 minutes to process the requests of 15 min.

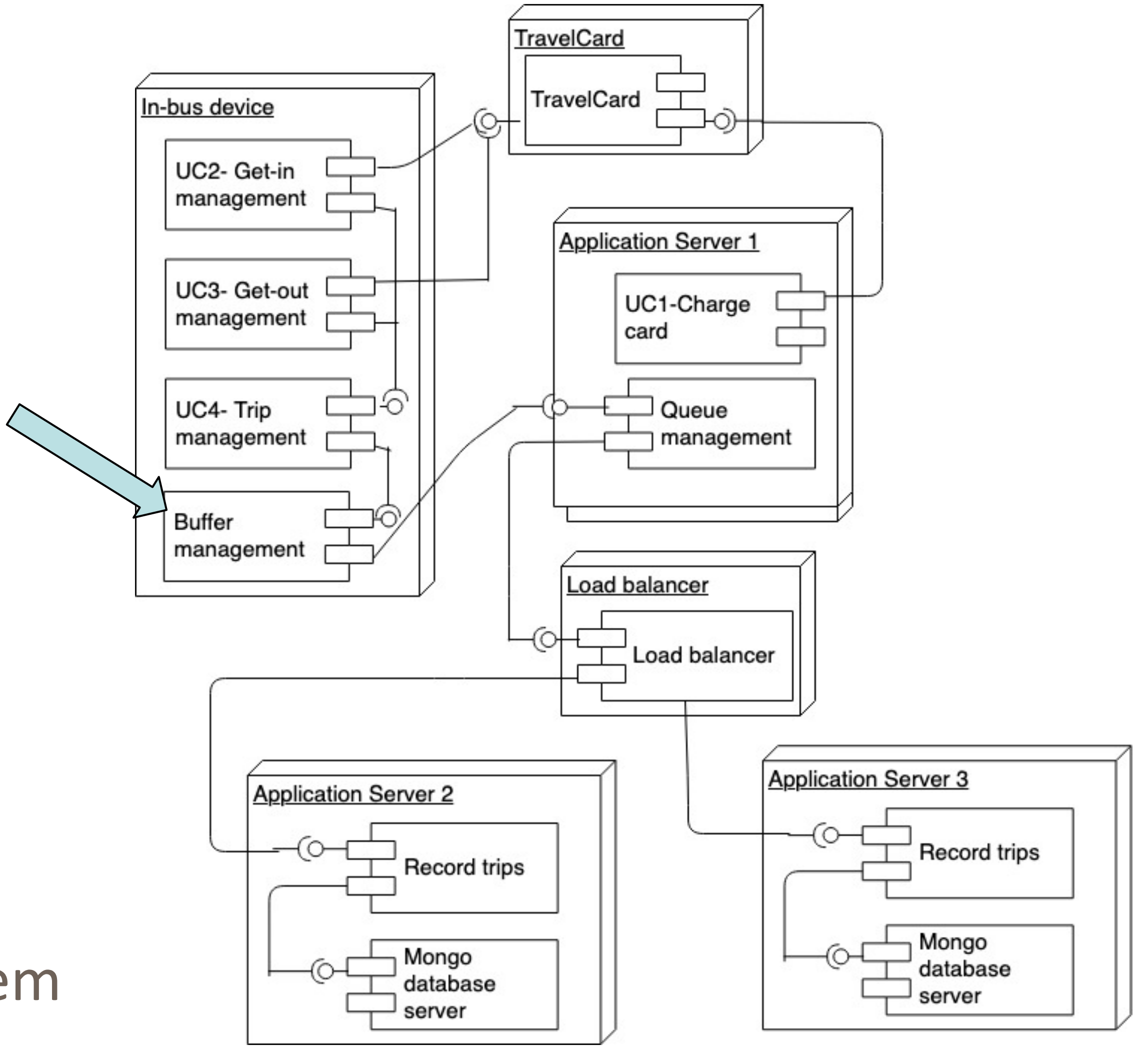
What is the impact of that on our queue capability?

Step 6 - Sketch Views and Record Design Decisions



UML communication diagram for the get-in use case

Step 6 - Sketch Views and Record Design Decisions



UML deployment diagram for the system

Step 7 - Perform Analysis of Current Design

Completely addressed

- All use cases
- QA1
- QA4
- QA2

Partially addressed

No addressed

- ~~QA2~~
- QA3
- QA5

Step 8 - Iteration If Necessary

The current sketches do not address the quality attributes:

- QA3 - Consider the specialization of the developers
- QA5 - The company expects to extend the system with data analytics capabilities

We stop here

(we decide to not address the two requirements)

Understanding the Architecture of the Banking System (Chapter 6)

The goal of the project is to digitally sign and send 2 million bank statements each month to the signature provider.

The goal of the new release is to develop a user interface to manage the system.

Understanding the Architecture of the Banking System

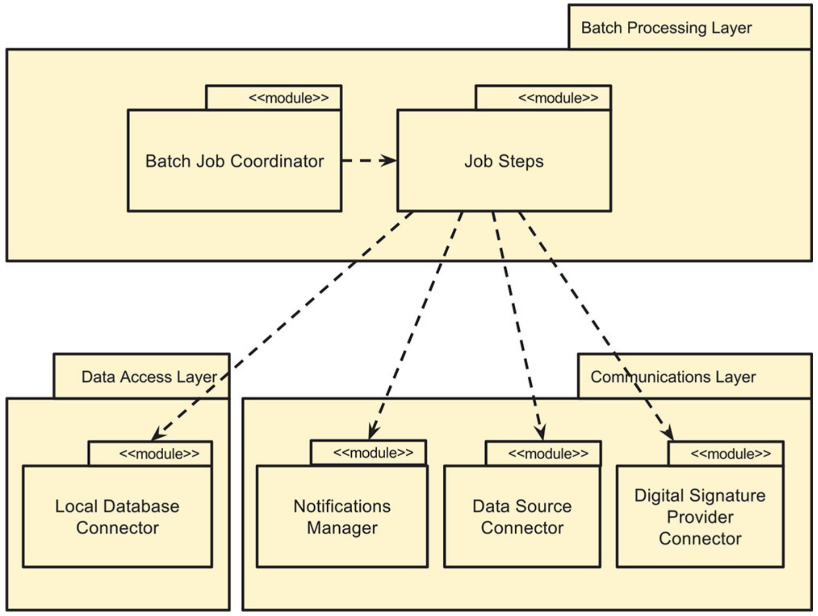
- Use cases for the Banking System:
    1. Reprocessing a number of statements
    2. Log in
    3. Generate reports regarding the process
    4. The administrator queries the log to display the activities of a user or group of users

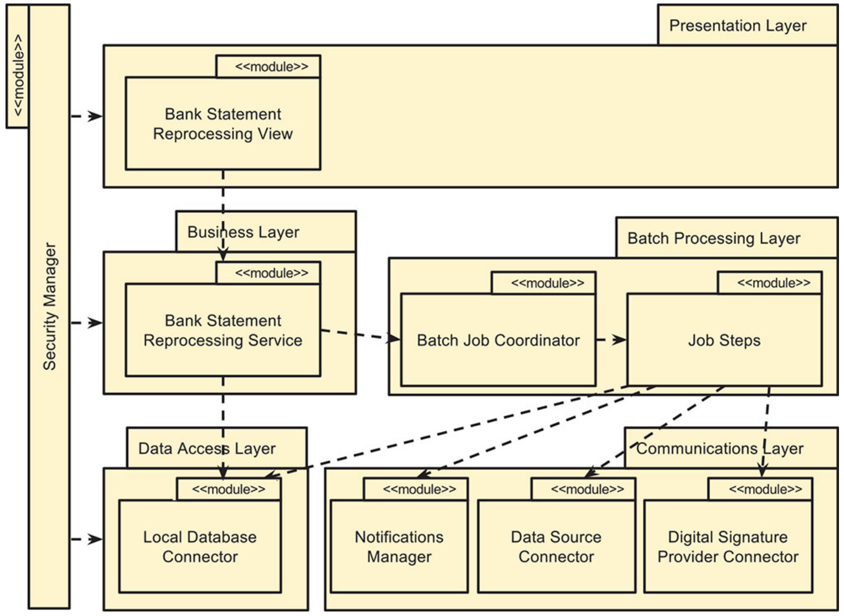
Understanding the Architecture of the Banking System

- What is the reference architecture of the original software?
- What is the proposed reference architecture for the extension?
- Why did the architect change the used reference architecture?

Understanding the Architecture of the Banking System

- What are the differences between the two diagrams below?
- The names of the layers are different from the classic data-business-presentation structure. Is the use of layers pattern correct?





Understanding the Architecture of the Banking System

- Would you agree with the choice of Spring Security for authorization and authentication? (Page 153)

Understanding the Architecture of the Banking System

- Can the application run several instances of the batch tool in parallel?
- Does the use of Spring have impact on the use of the existing batch tool?

Understanding the Architecture of the Banking System

Quality attributes for the banking System:

1. Under normal operating conditions, the batch process is executed in its entirely 100% of the time
2. Under normal operating conditions, 2 million statements are read, processed, and send for signature in at most one hour
3. During normal process, the administrator receives notifications of failure. The program could be restarted and processes only non processed statements.
4. All users' operations are recorded in the log

Understanding the Architecture of the Banking System

Quality attributes for the FCAPS

1. Performance – Add a new server management protocol without changing the core components.
2. Availability–Management system resumes after fault in 30 sec.
3. Security – Record changes to the systems, i.e., who did what, and when?
4. Performance – The system collects performance data within 5 min with no losses.

Discussion: Understanding the Architecture of the Banking System

- What are the major differences between the architecture drivers of the FCAPS system and the Big Data system?
  - Performance
  - Scalability
  - Extensibility
  - Availability
  - Deployability

- What is the focus in both cases?

Understanding the Architecture of the Banking System

What are the impacts of these differences on the architectures of the systems?

Understanding the Architecture of the Big Data System (Chapter 5)

- The company provides popular content and online services to millions of users.
- The system should allow to use users log to troubleshoot services, provide management reports, support data analytics, detect anomalous behavior, and support investigating security and compliance issues.

Understanding the Architecture of the Big Data System

Quality Attributes for the FCAPS

1. Performance – Add a new server management protocol without changing the core components.
2. Availability–Management system resumes after fault in 30 sec.
3. Security – Record changes to the systems, i.e., who did what, and when?
4. Performance – The system collects performance data within 5 min with no losses.

Understanding the Architecture of the Big Data System

- (Performance) The system should collect 15000 events/second from 300 servers
- (Performance) Refresh real-time monitoring with less than 1 min latency
- (Performance) The execution time for emergency troubleshooting queries is less than 10 seconds
- (Performance) The execution time of ad hoc queries is less than 2 minutes
- (Scalability) The system should store troubleshooting raw data for at least two weeks
- (Scalability) The system should store raw data for 60 days
- (Scalability) The system should store per-minute aggregated data for 1 year
- (Extensibility) The system should allow adding new data source by just changing the system configuration
- (Availability) The system should operate with no downtime
- (Deployability) The deployment procedure should be fully automated
- Etc.

Understanding the Architecture of the Big Data System

What are the major differences between the QAs of the FCAPS and the Big Data Systems?

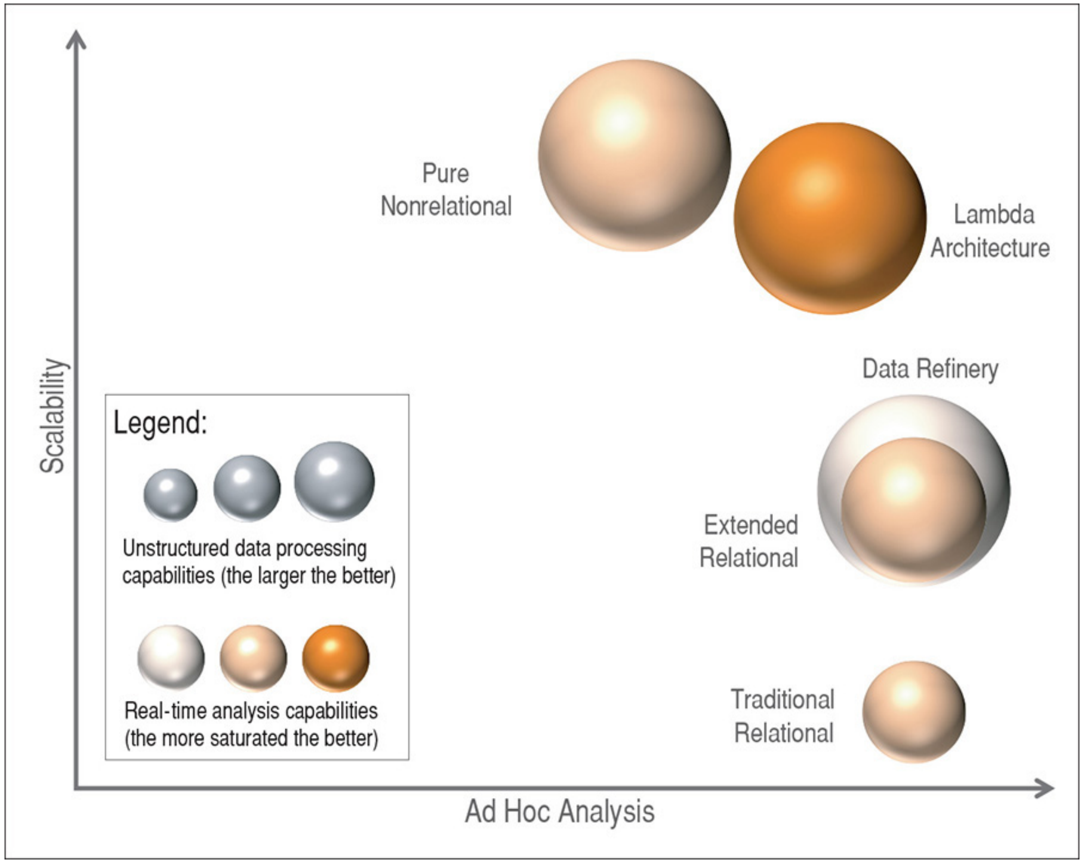
Understanding the Architecture of the Big Data System

Differences in Architectures

Relational database system is used in the FCAPS.

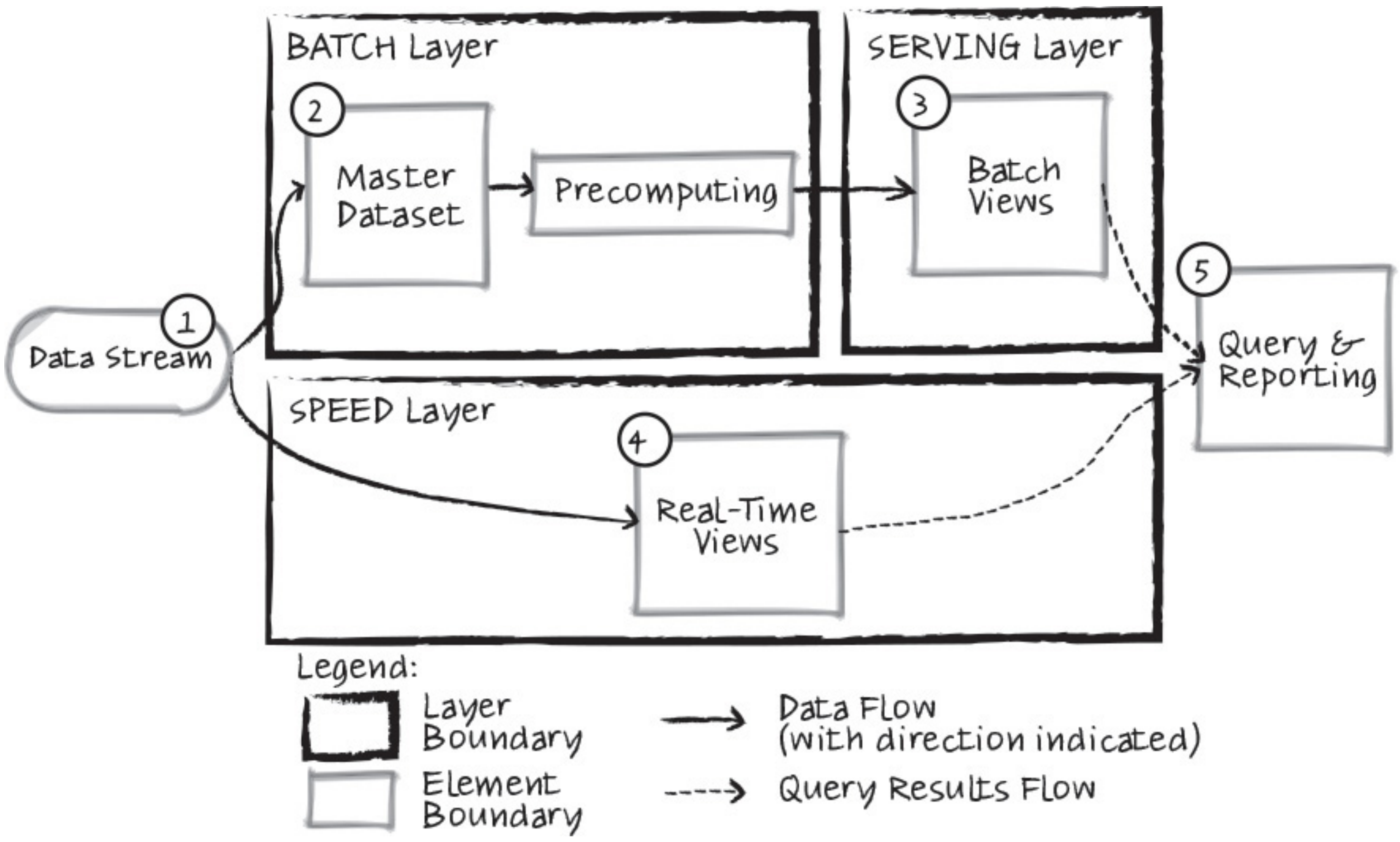
Lambda architecture is used in the Big Data System

Any comment?



Understanding the Architecture of the Big Data System

Why do we need three layers?



Thank you

Questions?