Software Engineering
Weeks 6.5 and 9: architecture & design

Lydie du Bousquet

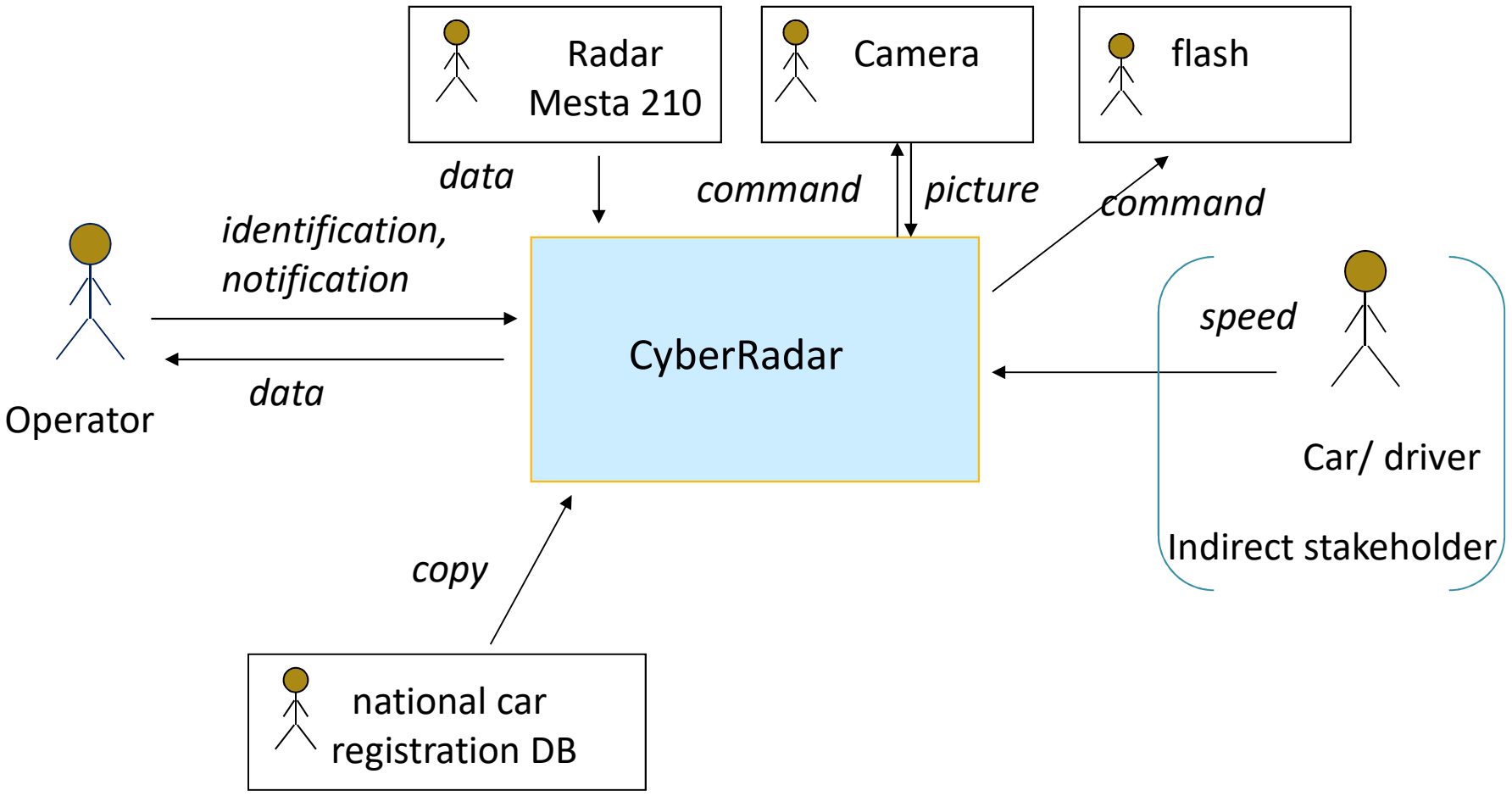
Lydie.du-bousquet@imag.fr

In collaboration with J.-M. Favre, I. Parissis, Ph. Lalanda, Y. Ledru

Radar system

- Identify the limits of what has to be developed
- Identify what the system is supposed to do
- Propose a first design for Radar system software

Example of Radar system



Radar system : Use cases

- Records a vehicle's speed
- Takes a photograph of the vehicle when it exceeds a threshold limit.
- Speeds camera, a high speed radar, camera, flashbulb
- Sends pictures and the related information to a management center
- Allows user to
  - Identify automatically the number plate and the owner of the car,
  - Check manually if the number plate is correctly identified
  - Validate manually the penalty document
  - Improve manually the picture
  - Fill the penalty document
  - Validate the penalty document

- Keeps data for a long time (?)
- Allows connection to national car registration DB

Radar system

- Why design is difficult in comparison with finding functionalities?

Software design: Why is it difficult?

- Inherent software complexity
- Requirements
- Regulatory constraints

Problem is complicated

- Team size, location and distribution

People

- Choices
  - Modules, organisation
  - Technical part

Decisions With Impacts

Software design: Why is it difficult?

- Inherent software complexity
- Requirements
- Regulatory constraints

What

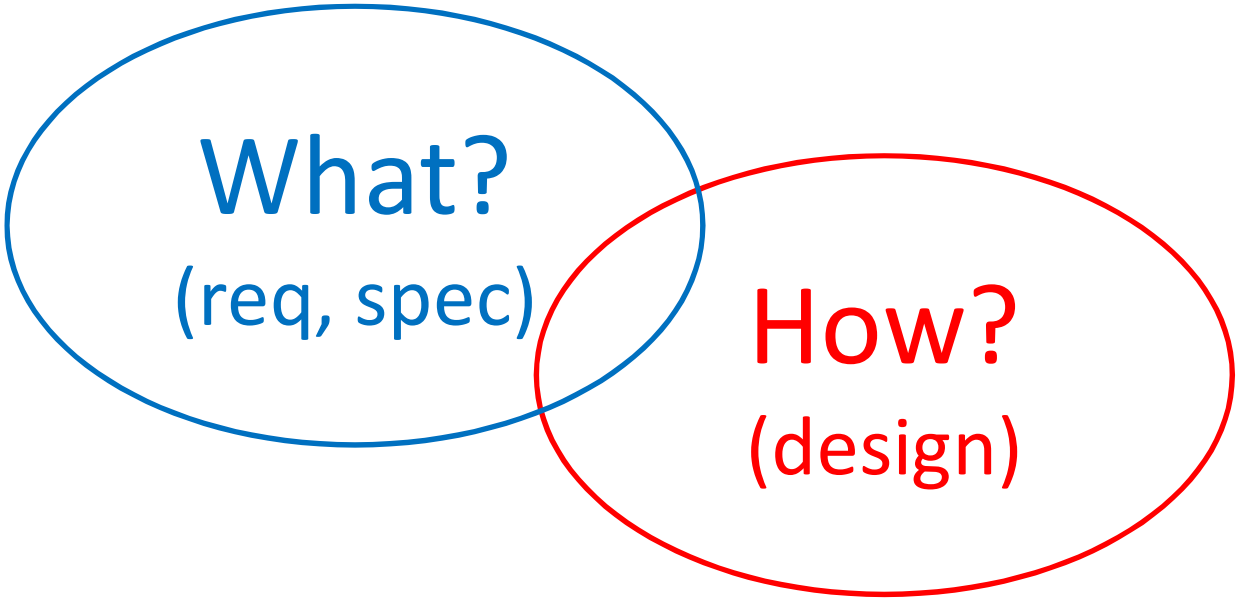
- Team size, location and distribution

Management

- Choices
  - Modules, organisation
  - Technical part

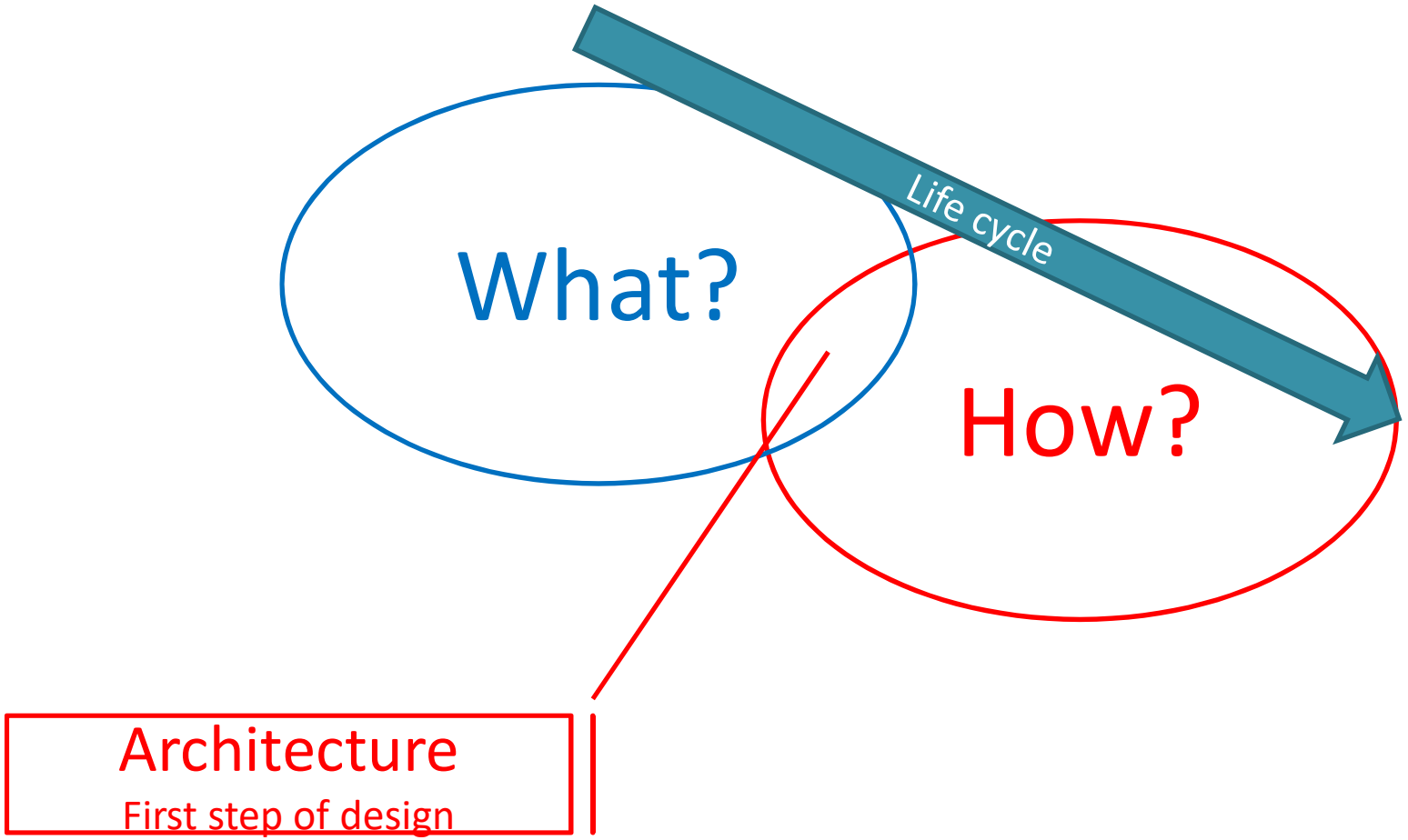
How?

Software design: Why is it difficult?



http://www.slideshare.net/arslantumbin/software-architecture-vs-design

Did you say architecture?



http://www.slideshare.net/arslantumbin/software-architecture-vs-design

Software architecture

- Refers to the high level structures of a software system
- Discipline of
  - creating such structures, and
  - documenting of these structures

- Advantages
  - facilitates communication between stakeholders,
  - captures early decisions about the high-level design,
  - allows reuse of design components between projects

Schedule

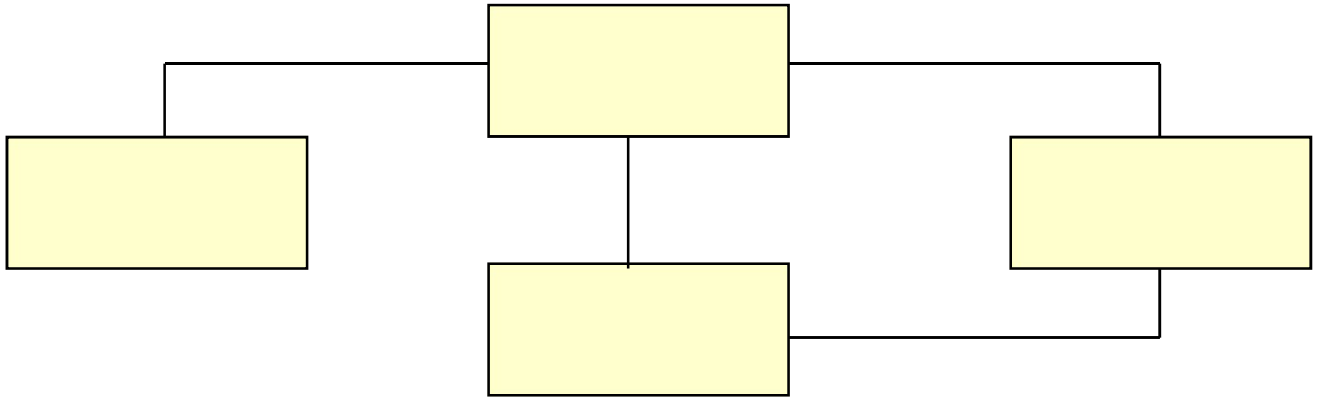
- Software architecture
  - Design
  - Representation

Software design: Solutions? Some directions

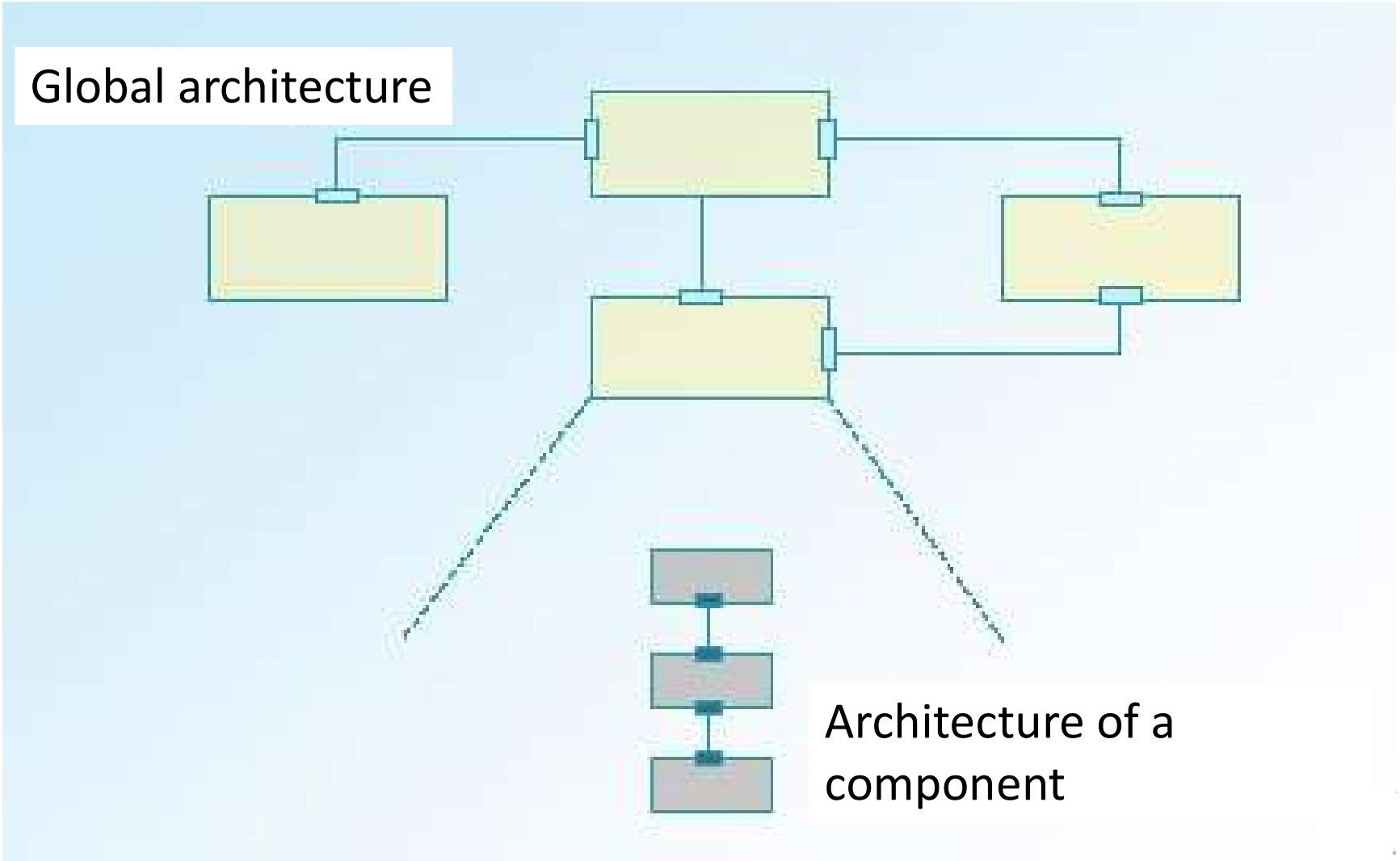
- Deal with complexity
  - Divide and conquer
  - Abstraction
  - Separation of preoccupation
- Be rigorous
- Use “tools”
  - Intellectual tool = methods

First level of design

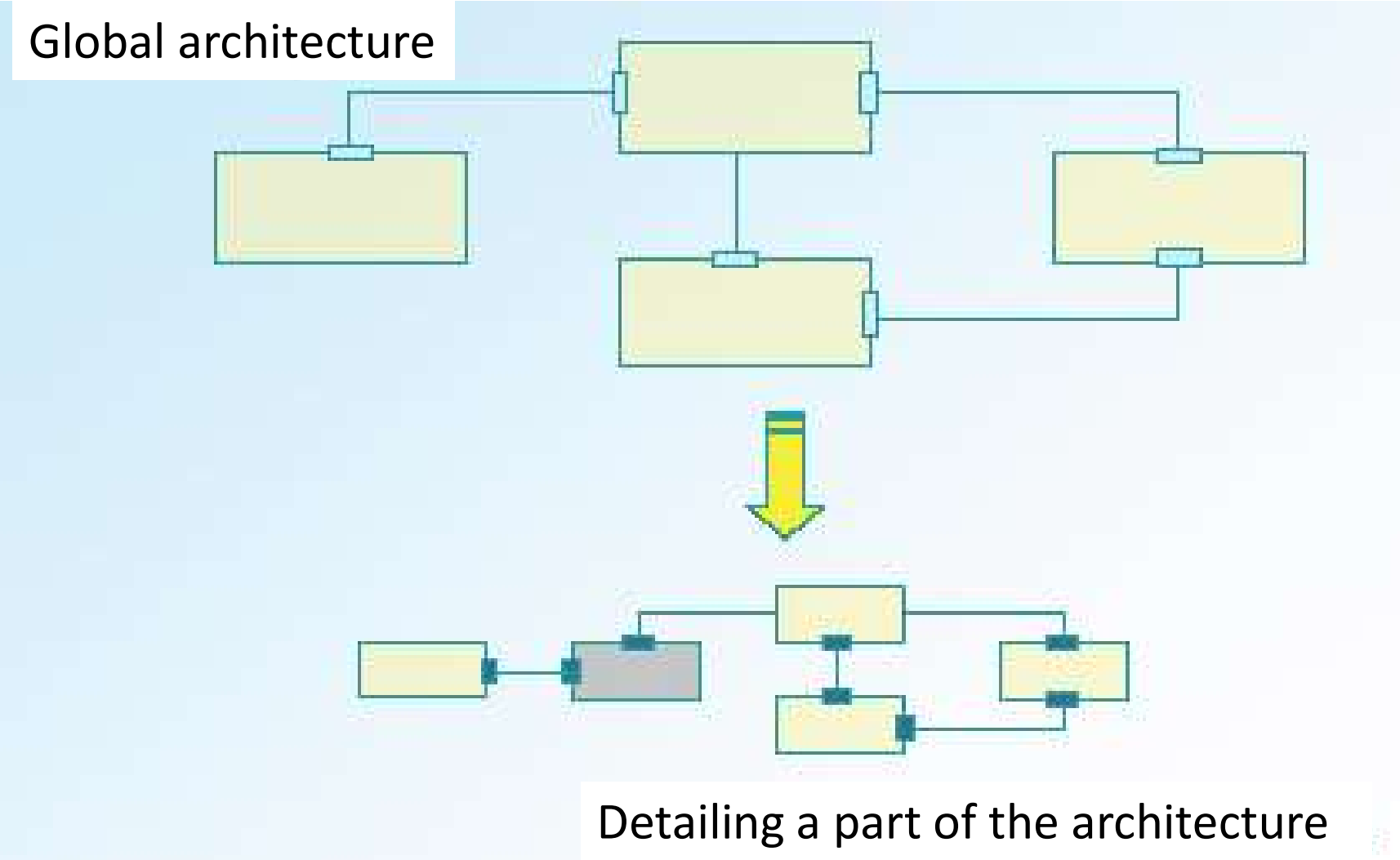
- Propose a first decomposition of the problem/solution
  - Find a set of abstract components
  - Organize them
  - Check relevance
- Then detail



Detail: a component is refined



Detail: add technical component and/or specify the interfaces



Architecture representation
How?

- Using abstraction, separation of concerns
- With some UML views to capture
  - the boundaries
  - the functionalities
  - the structure
  - the behaviors
  - the physical repartition

Architecture representation
How?

- **Boundaries of the system**
  - **UML context diagram**

- **Functionalities**
  - **Use case** diagram

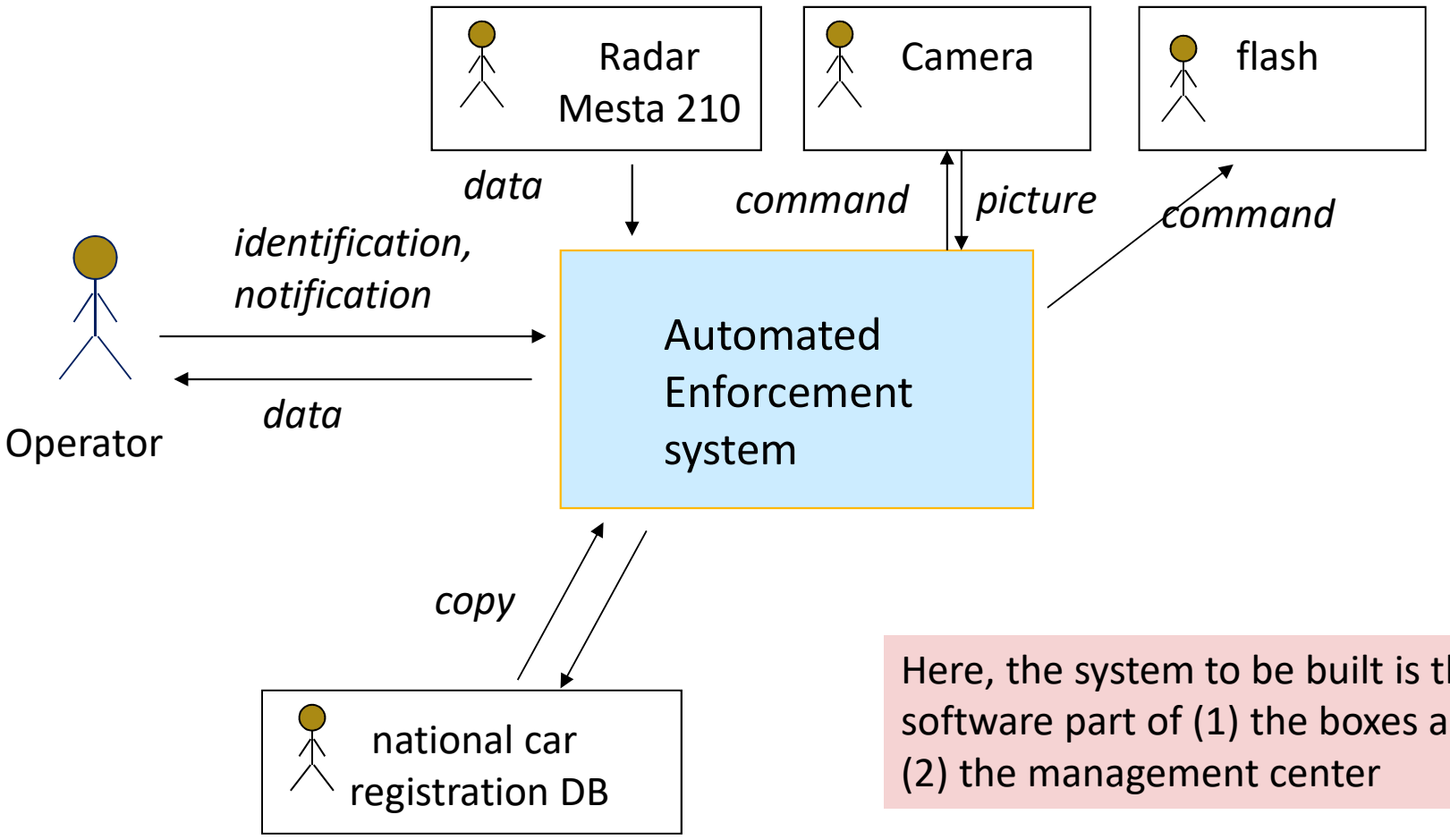
Architecture representation
How?

- The **structure** at a high-level of abstraction
  - **« Logical views »**
  - **Component** diagram (or simplified class diagram)
- Some example of the **behaviors**
  - **« Dynamical views »**
  - **Sequence** diagrams
- The repartition on the **machines**
  - **« Physical views »**
  - **Deployment** diagram

Context diagram

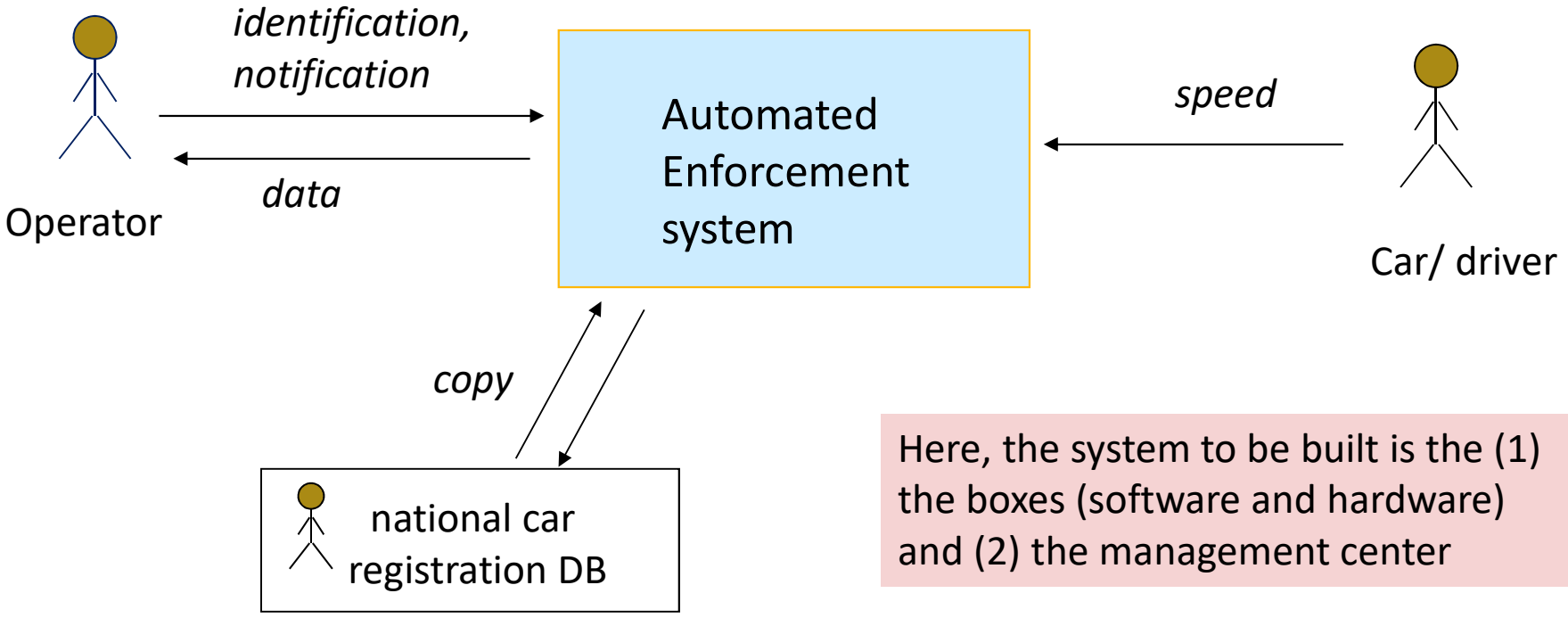
- To describe the **boundaries** of the system to develop
- The system is represented as a whole
- Each element interacting with the system is identified with an **actor**
  - Human or external system
  - That uses or is used by the system to develop

Example of Radar system



Here, the system to be built is the software part of (1) the boxes and (2) the management center

Example of Radar system



Here, the system to be built is the (1) the boxes (software and hardware) and (2) the management center

Use cases

- To list the functionalities of the systems
- Don’t forget to express the constraints
- They will be used to elaborate a set of components

Example of Radar system

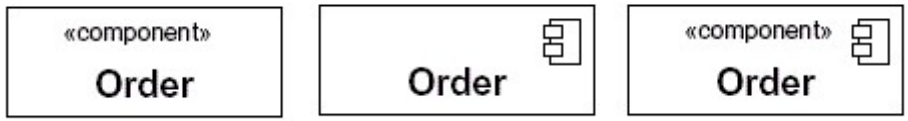
- Records a vehicle's speed
- Takes a photograph of the vehicle when it exceeds a threshold limit.
- Speeds camera, a high speed radar, camera, flashbulb
- Sends pictures and the related information to a management center
- Allows user to
  - Identify automatically the number plate and the owner of the car,
  - Check manually if the number plate is correctly identified
  - Validate manually the penalty document
  - Improve manually the picture
  - Fill the penalty document
  - Validate the penalty document

- Keeps data for a long time (?)
- Allows connection to national car registration DB

Logical views

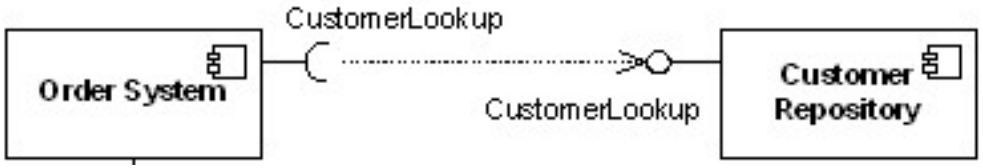
Structure of the application

- Simple class diagram OR
- Simple component diagram



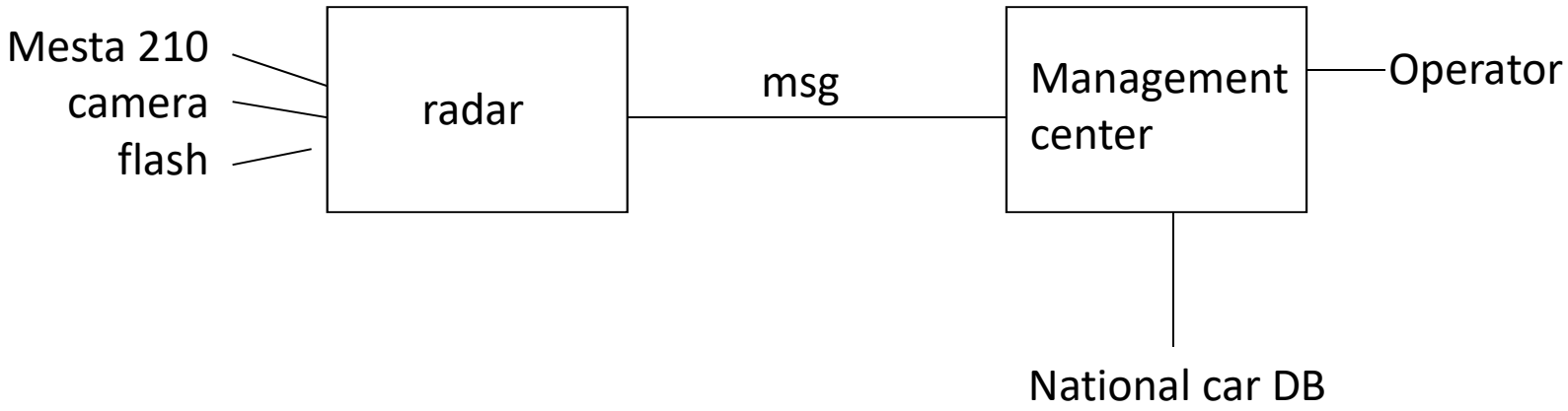
Component

Interfaces



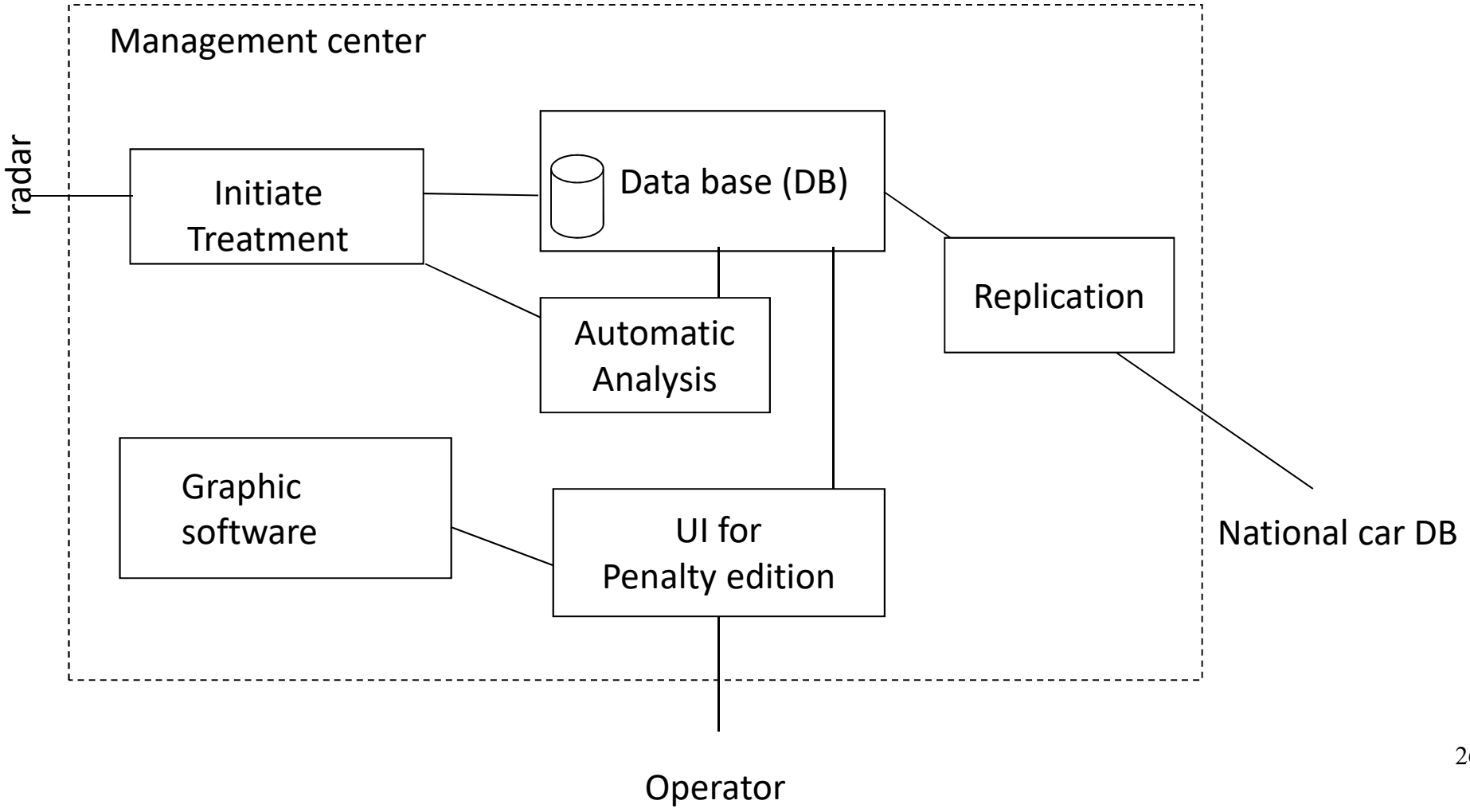
Relations

Example of Radar system
Global logical view



NB: For the moment, we do not want to focus on interface (abstraction);
So interface are not represented =>
The component diagram looks like a simplified class diagram

Example of Radar system
Focus on Management center
One solution among several



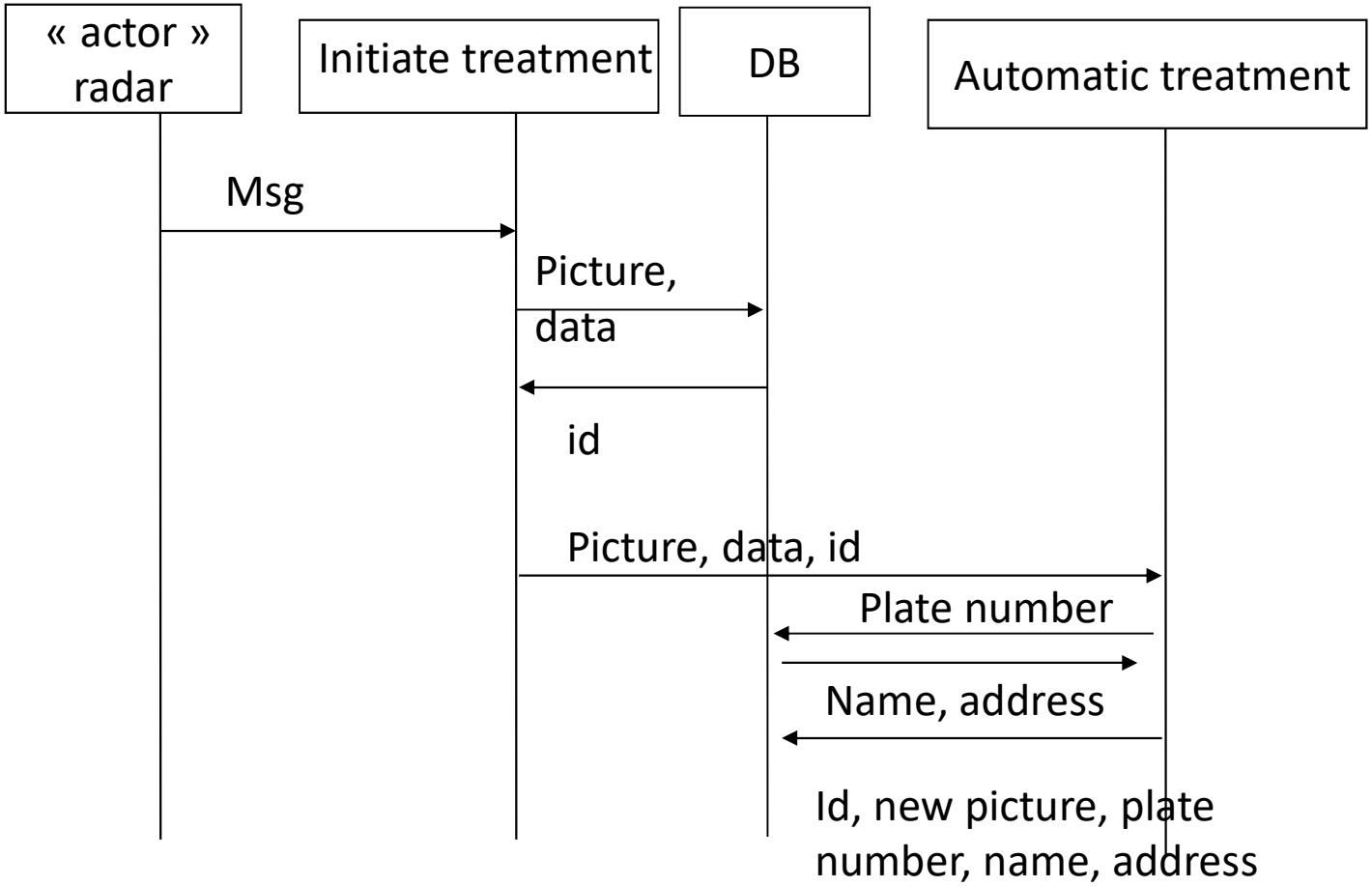
Dynamical Views

to illustrate the system behaviors

- Choose relevant behaviors (not all)
- Describe them with sequence diagrams

Example of Radar system

Messages are recieved, registered and automatically treated



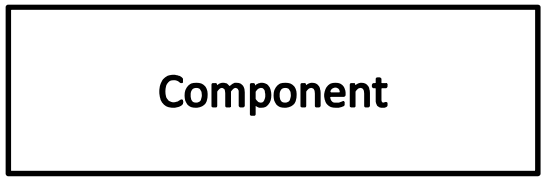
Example of Radar system

Operator asks a new picture, treats it (automatic treatment failed)
fills penalty document

Physical view to represent the repartition of the component

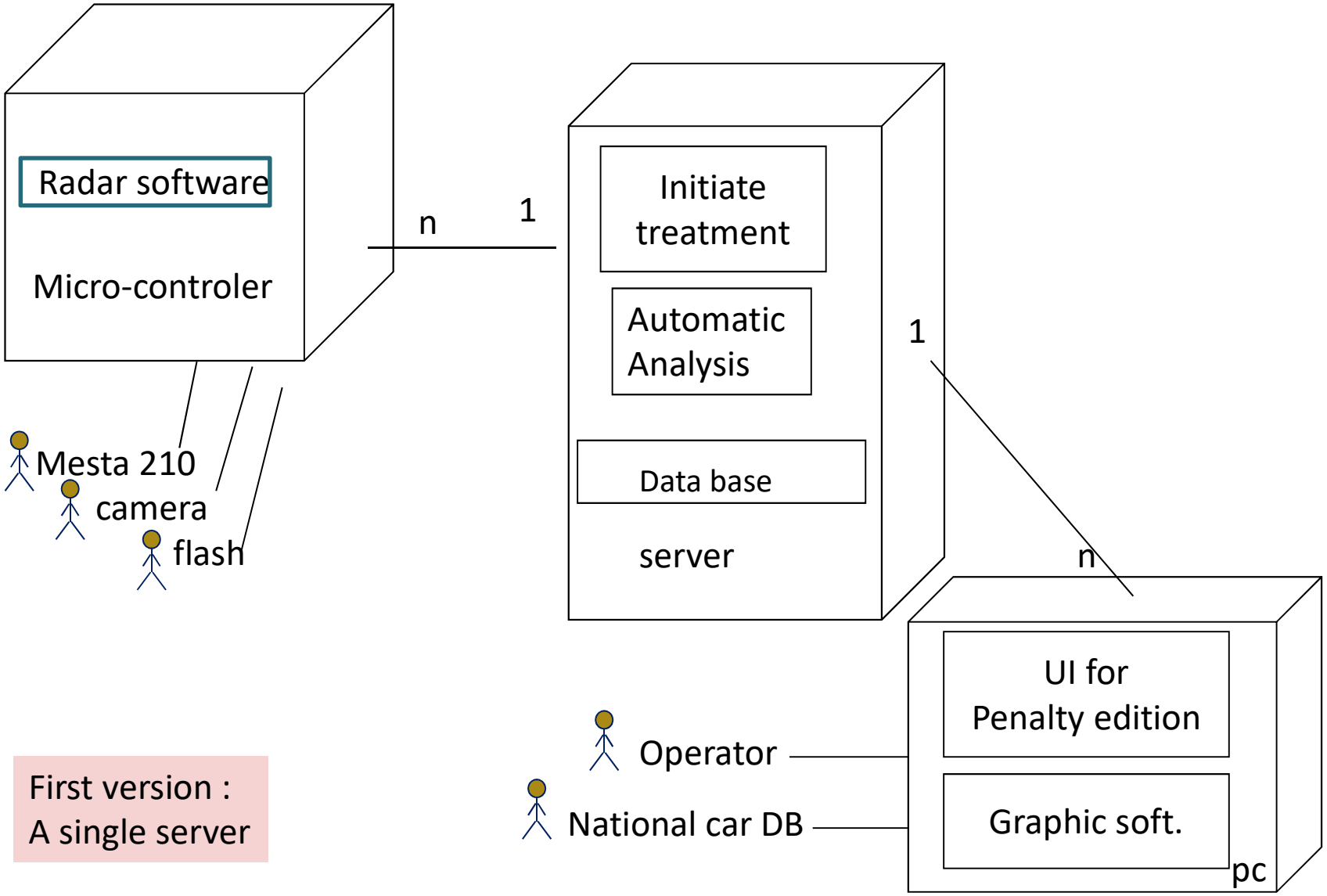
- Use a simplified UML deployment diagrams
- Machines (devices)
  are represented as « cube »

- Components
  are represented as usal



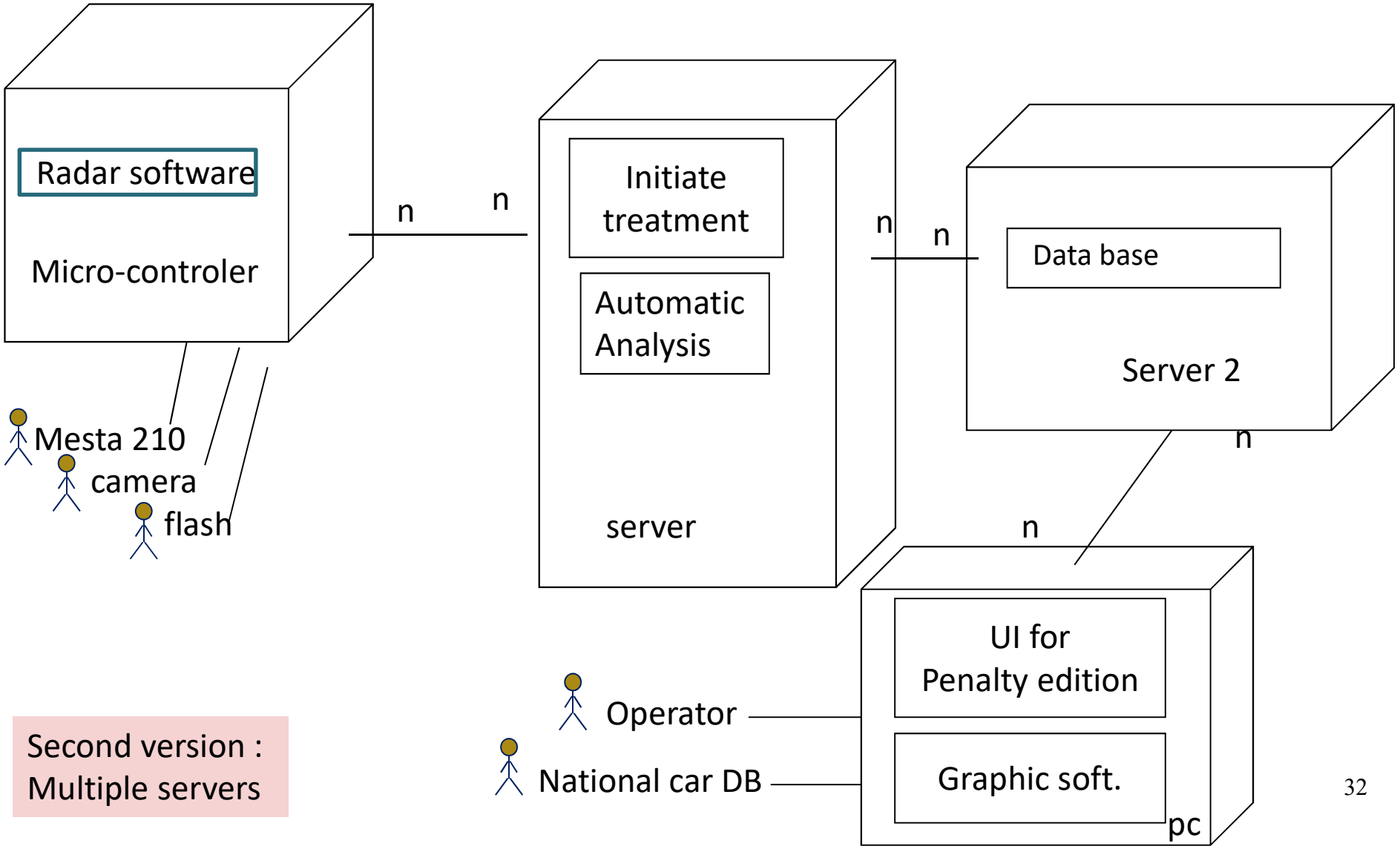
- Specify relation between machines
  with cardinalities

Example of Radar system



First version :
A single server

Example of Radar system



Second version :
Multiple servers

Architecture and difficulties

- Architecture = first step(s) of conception
- You are in the process of **choosing a solution**
  - Choosing a set of components
  - Choosing how to connect them
- That is why it is difficult
- Any methods?
  - Is there any universal method to solve a problem?

Architecture design
some clues

- Identify a set of components
  - Method CBSP (University of Southern California)
  - Organize requirement into thematic groups
  - Make components emerging from these groups
  - Use abstraction if necessary
- Organize the components
  - As you can
- Check

Architecture design
some clues

- Identify a set of components
  - Method CBSP (University of Southern California)
  - Organize requirement into thematic groups
  - Make components emerging from these groups
  - Use abstraction if necessary
- Organize the components
  - **As you can** or **with architectural styles**
- Check

Architecture design
some clues

- To organize the component, it is possible to use “architectural styles” (aka patterns)
- Architectural patterns are classical solutions for component organization
  - MVC (model-view-controller)
  - Client-server
  - 3 tiers
  - Layers…

Architectural patterns

Architectural patterns - 1

- Solution to a classical problem
  - Abstract
  - Well-known behaviors
  - Well-known advantages and drawbacks
- Helps
  - Starting the architectural design
  - Communication among stakeholders
  - Evaluation of the architecture

Architectural patterns - 2

- More and more important
- Emerging catalogues
  - Not well organized
  - With different levels of abstractions
- Example of architectural pattern families / style
  - data-flow
  - data-centered
  - hierarchical
  - for distributed architectures
  - for UI

Catalog of architectural patterns

- Three-tier
- Multilayered architecture
- Model-view-controller
- Domain Driven Design
- Micro-kernel
- Blackboard pattern
- Sensor-controller-actuator
- Presentation–abstraction–control

Catalog of architectural styles [ed

Structure [edit]

- Component-based
- Monolithic application
- Layered
- Pipes and filters

Shared memory [edit]

- Database-centric
- Blackboard
- Rule-based

Messaging [edit]

- Event-driven aka implicit invocation
- Publish-subscribe
- Asynchronous messaging

Adaptive systems [edit]

- Plug-ins
- Microkernel
- Reflection

Wikipédia/wikiwand

Category	Architecture styles
Communication	Service-Oriented Architecture (SOA), Message Bus
Deployment	Client/Server, N-Tier, 3-Tier
Domain	Domain Driven Design
Structure	Component-Based, Object-Oriented, Layered Architecture

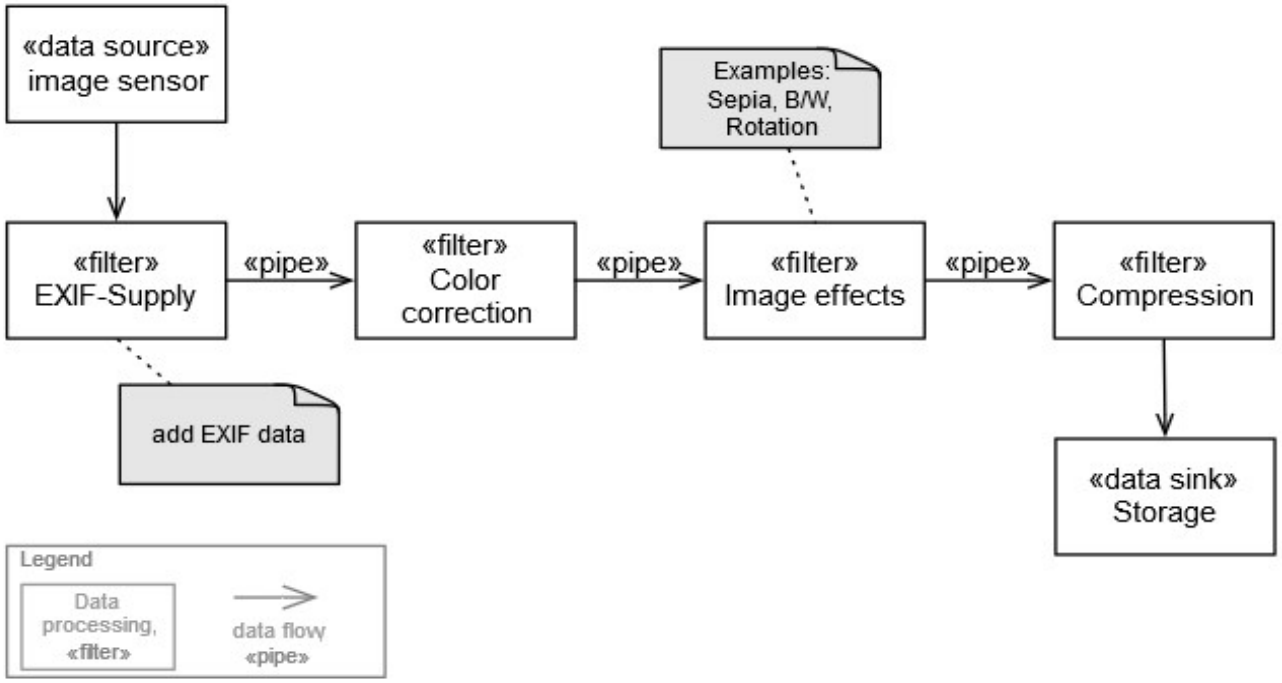
https://msdn.microsoft.com/

1. **Layered pattern**
2. **Client-server pattern**
3. **Master-slave pattern**
4. **Pipe-filter pattern**
5. **Broker pattern**
6. **Peer-to-peer pattern**
7. **Event-bus pattern**
8. **Model-view-controller pattern**
9. **Blackboard pattern**
10. **Interpreter pattern**

https://medium.com/

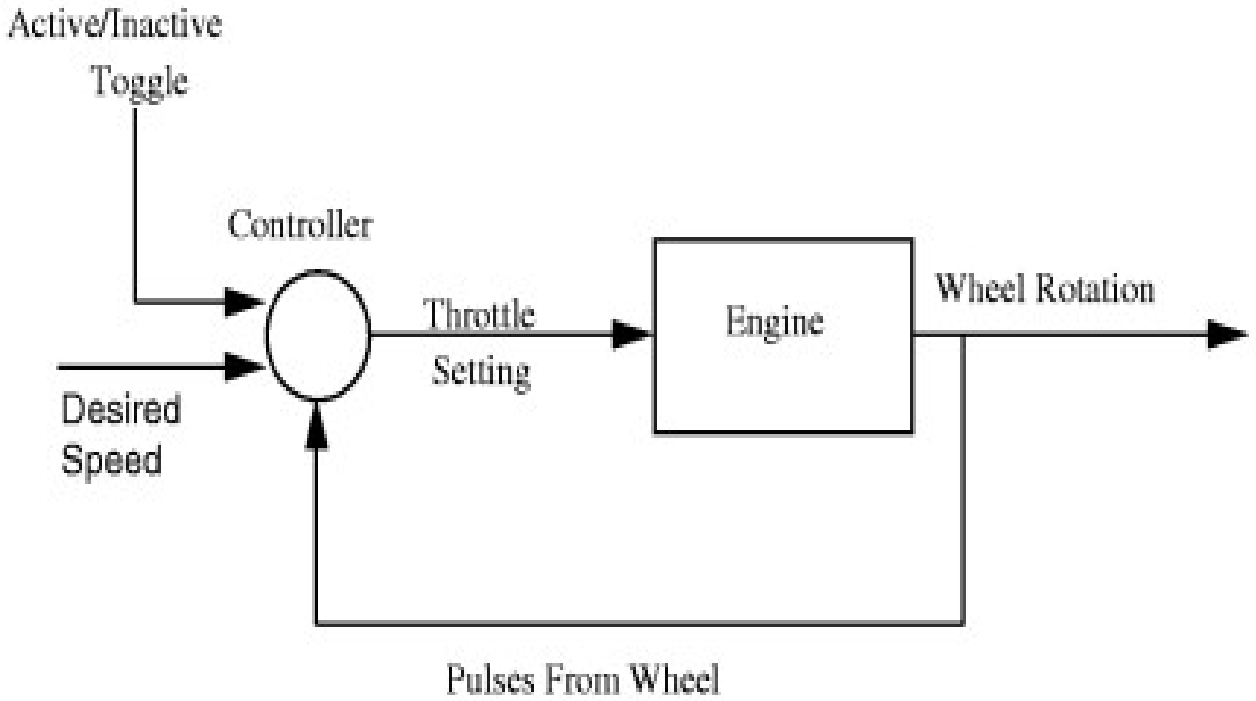
Data-flow style example: Pipes and filters

- Architecture is organized as a set of transformations
  - Filters are the components dedicated to transformations
  - Pipes are dedicated to the communication



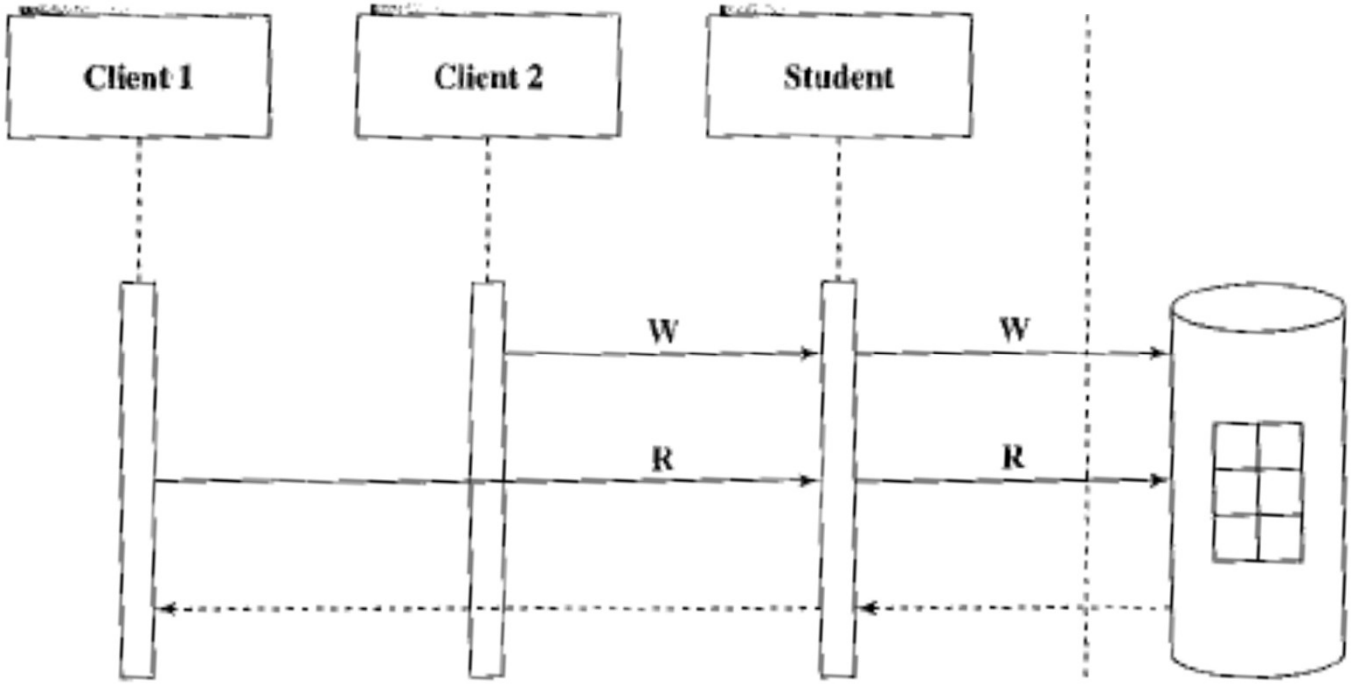
Data-flow style example: Process-Control Architecture

- For systems that have to
  - maintain an output to a specific value
  - reach a specific objective



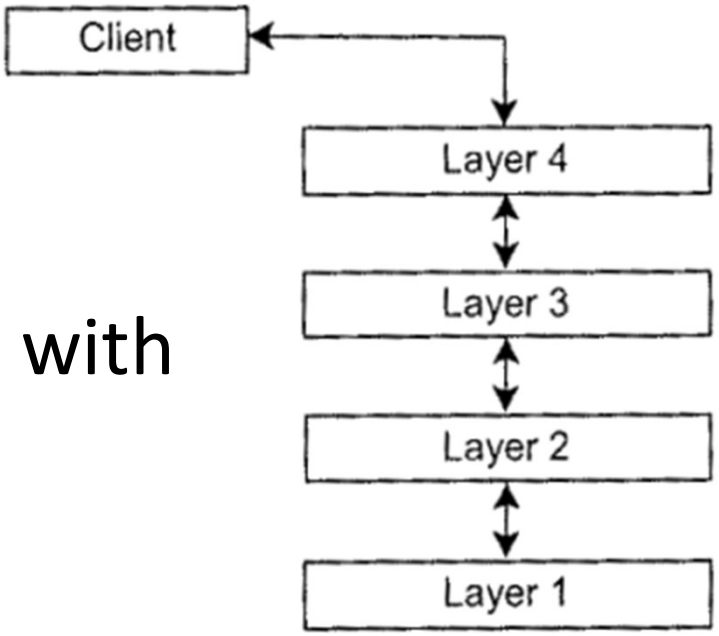
Data-centered style example: Repository-style

- Architecture is organized around a repository (data-base)



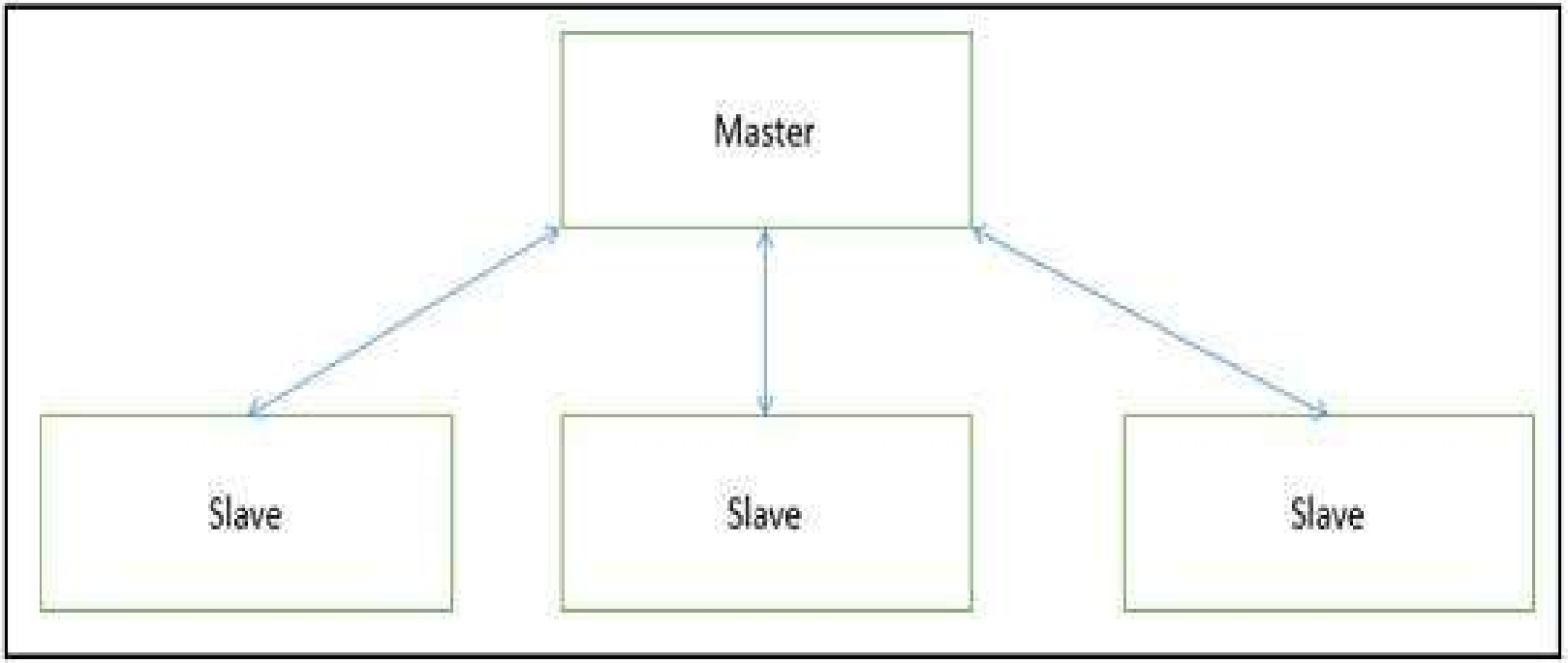
Hierachical style example: Layered

- Components are organized into layers
- Each layer deals with a **level of abstraction**
- Each layer communicates with its **immediate neighbors**



Hierarchical style example: Master-slave

- A component (master) controls the execution of the others (slaves)



Distributed architecture style example Client-server (two-tier)

- Involve a separate client and server system, connecting through a network.

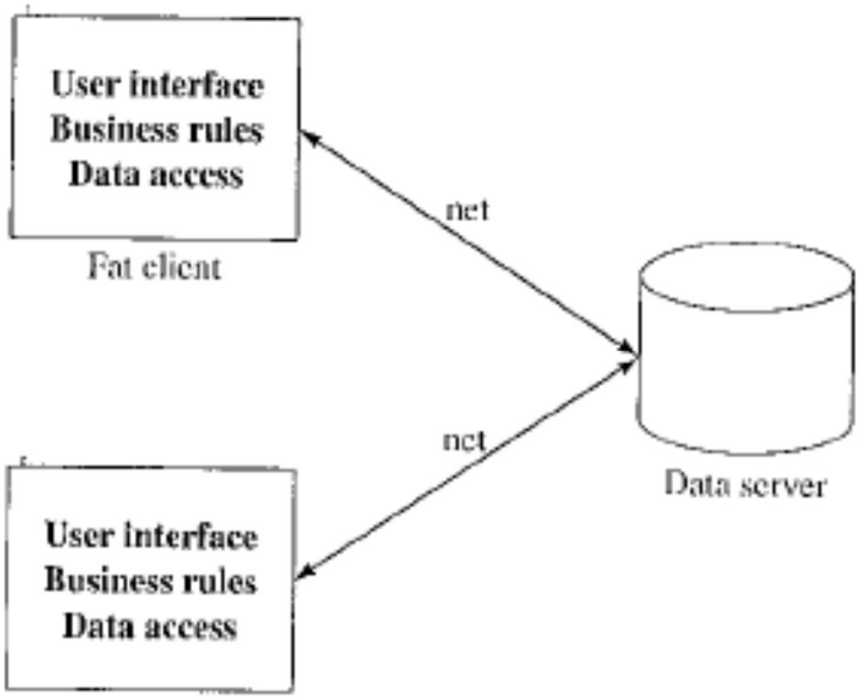
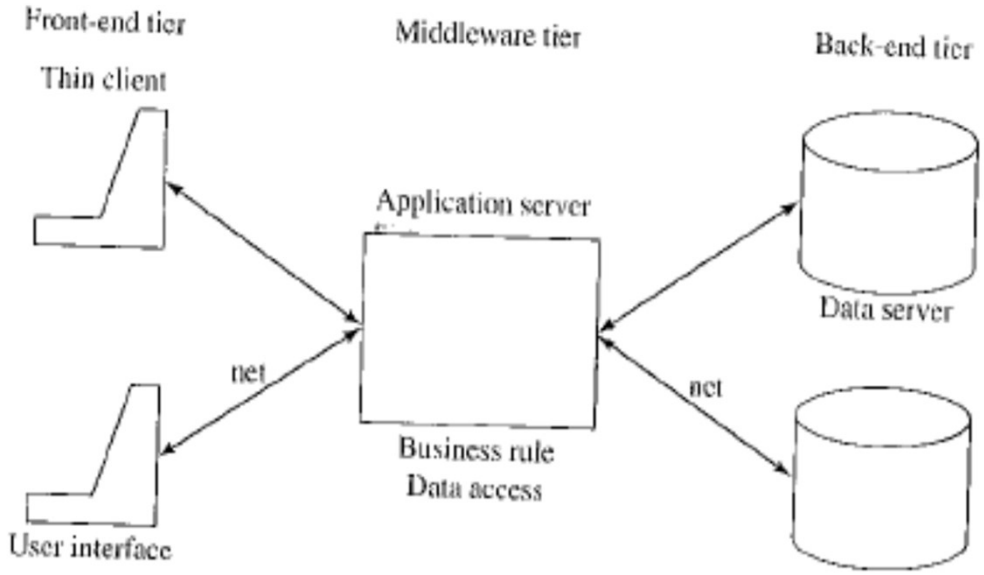


Figure 10.1
Two-tier client-server architecture

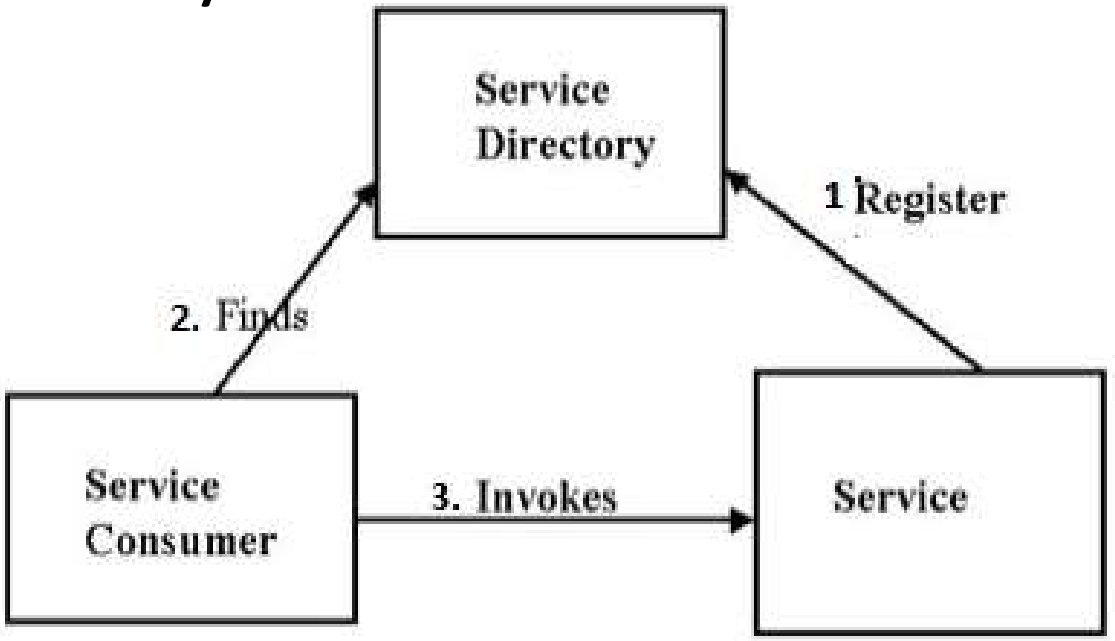
Distributed architecture style example three-tiers

- Separation of **functionality** into segments
  (different from abstraction layers)
- Segment (tier) can be located on a physically separate computer.

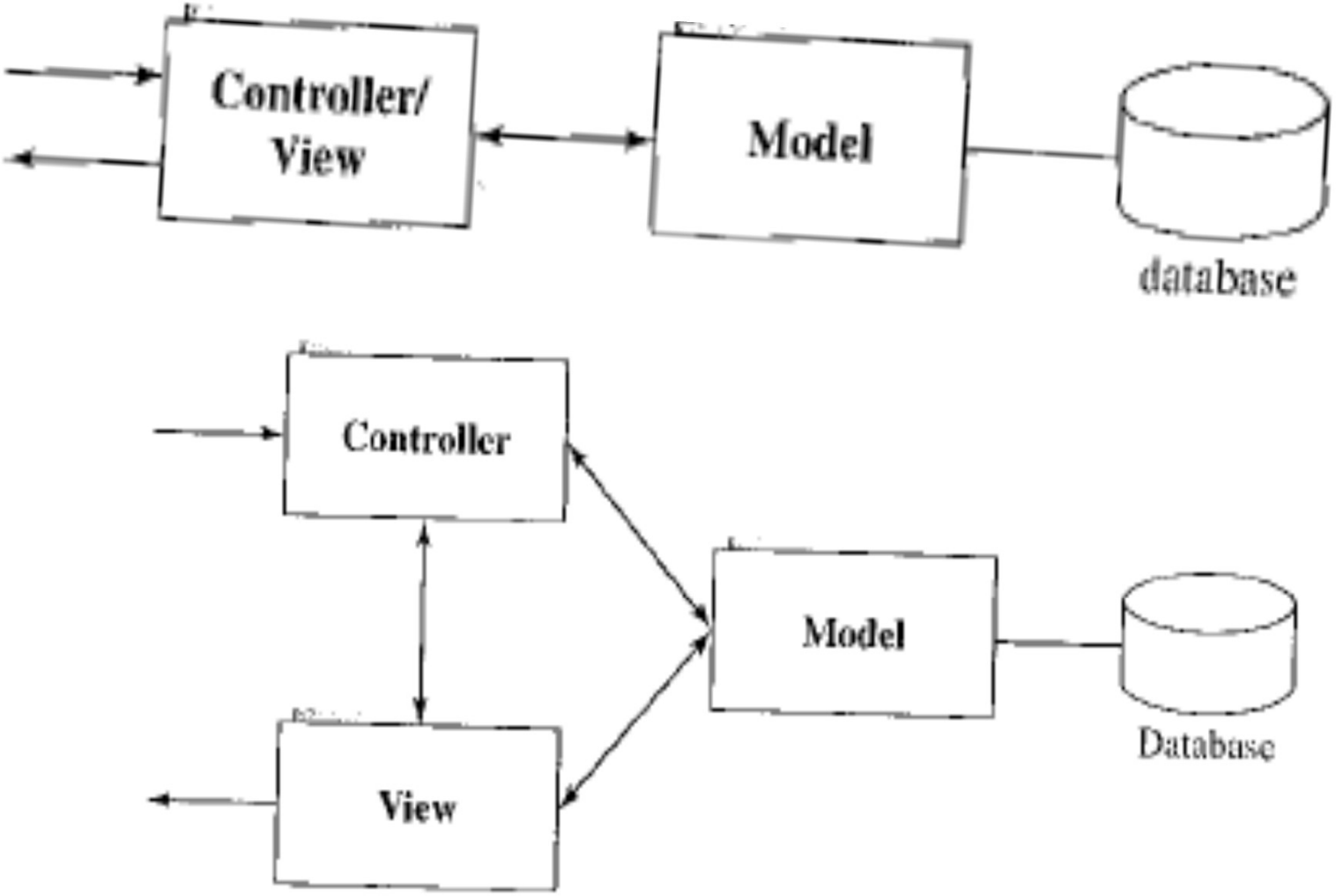


Distributed architecture style example Service-oriented architecture

- Functionalities embedded in “services”
- Available services are “published”
- A service looks for what it needs through the “service directory”

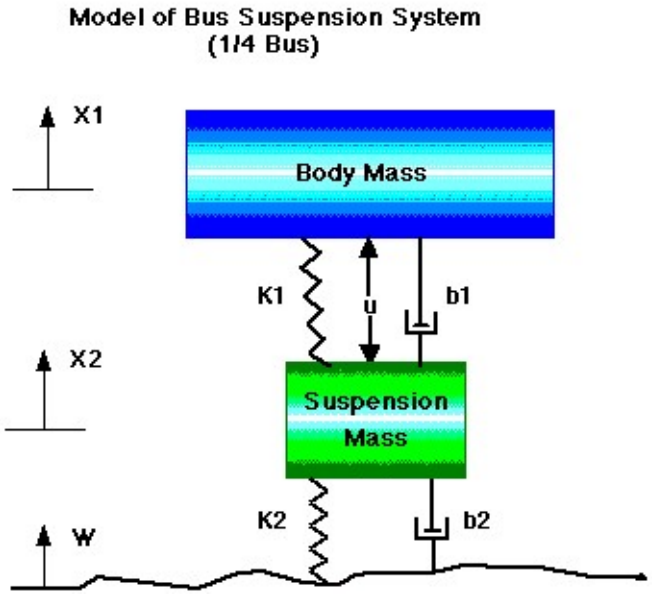


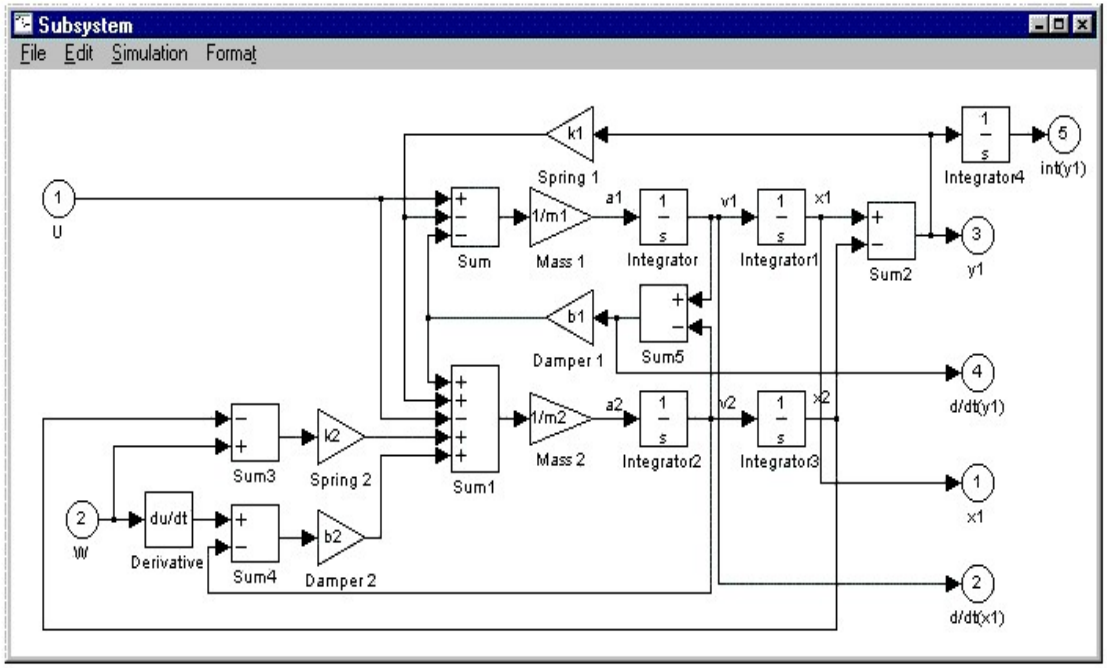
UI architectural style example
MVC



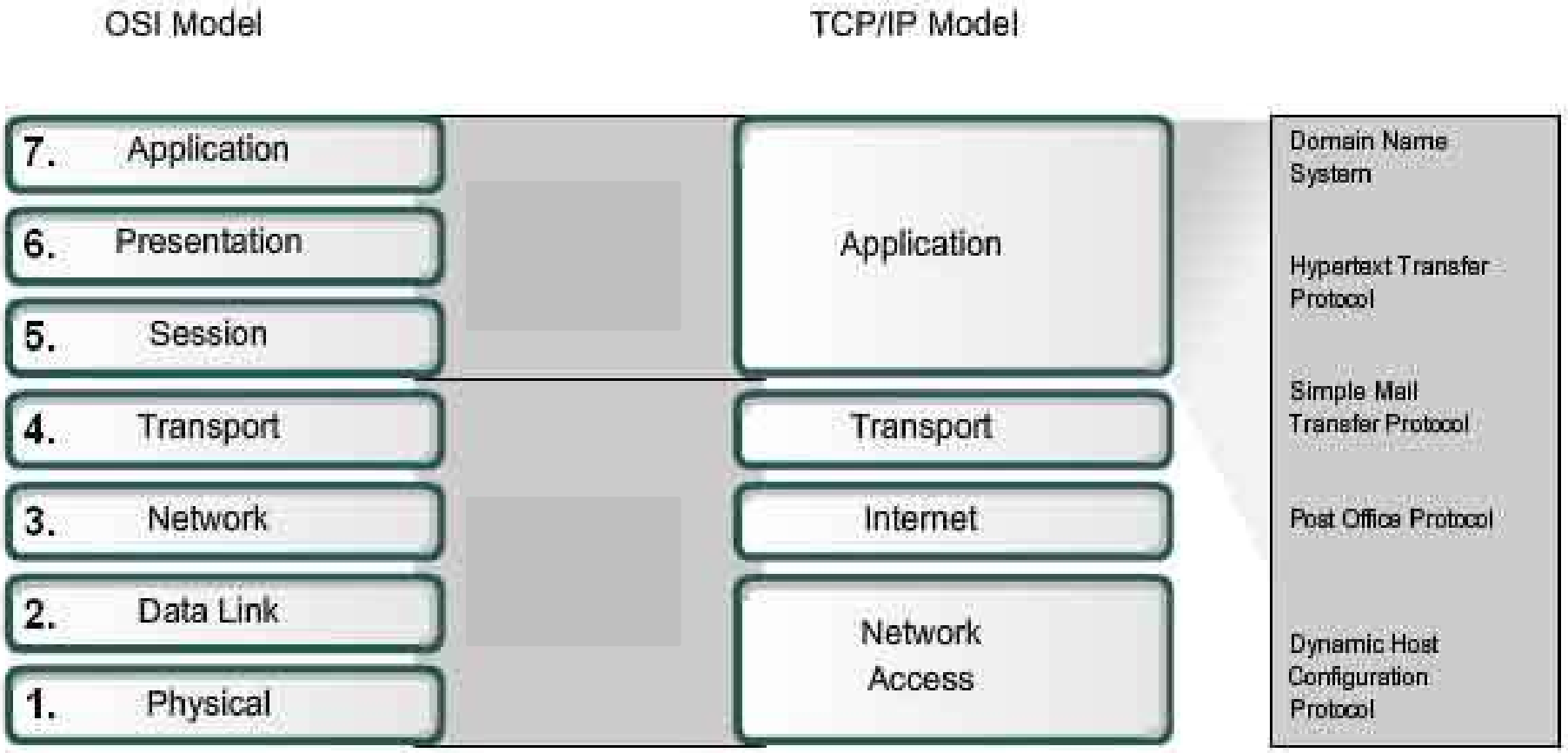
Can you recognize the following architectural styles?

Bus suspension





Communication protocole



52

health e waterways
Integrating knowledge

EHMP Estuarine/Marine

EHMP Freshwater

EHMP Event Monitoring

Models

Management Actions

SEQ Water

Bureau of Meteorology

Landuse

Distributed Databases

Health-e-Waterways Web Portal

SECURITY LAYER

Users

General Public

State Government

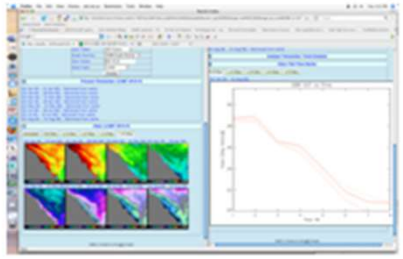
Local Governments

Water Resource Managers

Researchers

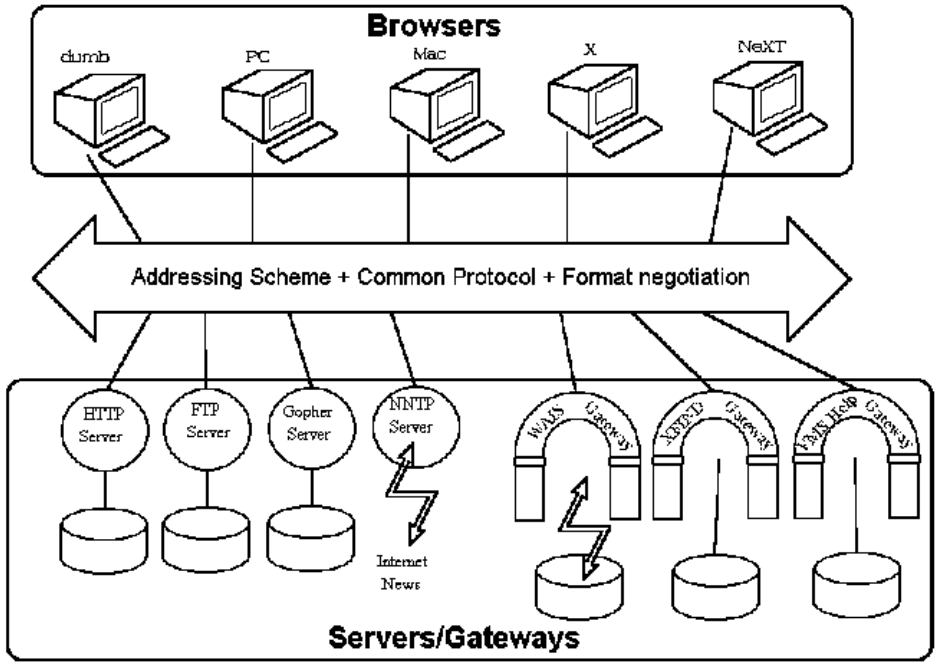
Scientists

Hydrologists



Example : WWW

- Users can access the information from the WWW which is the front end software supported for on-line retrieval of information.

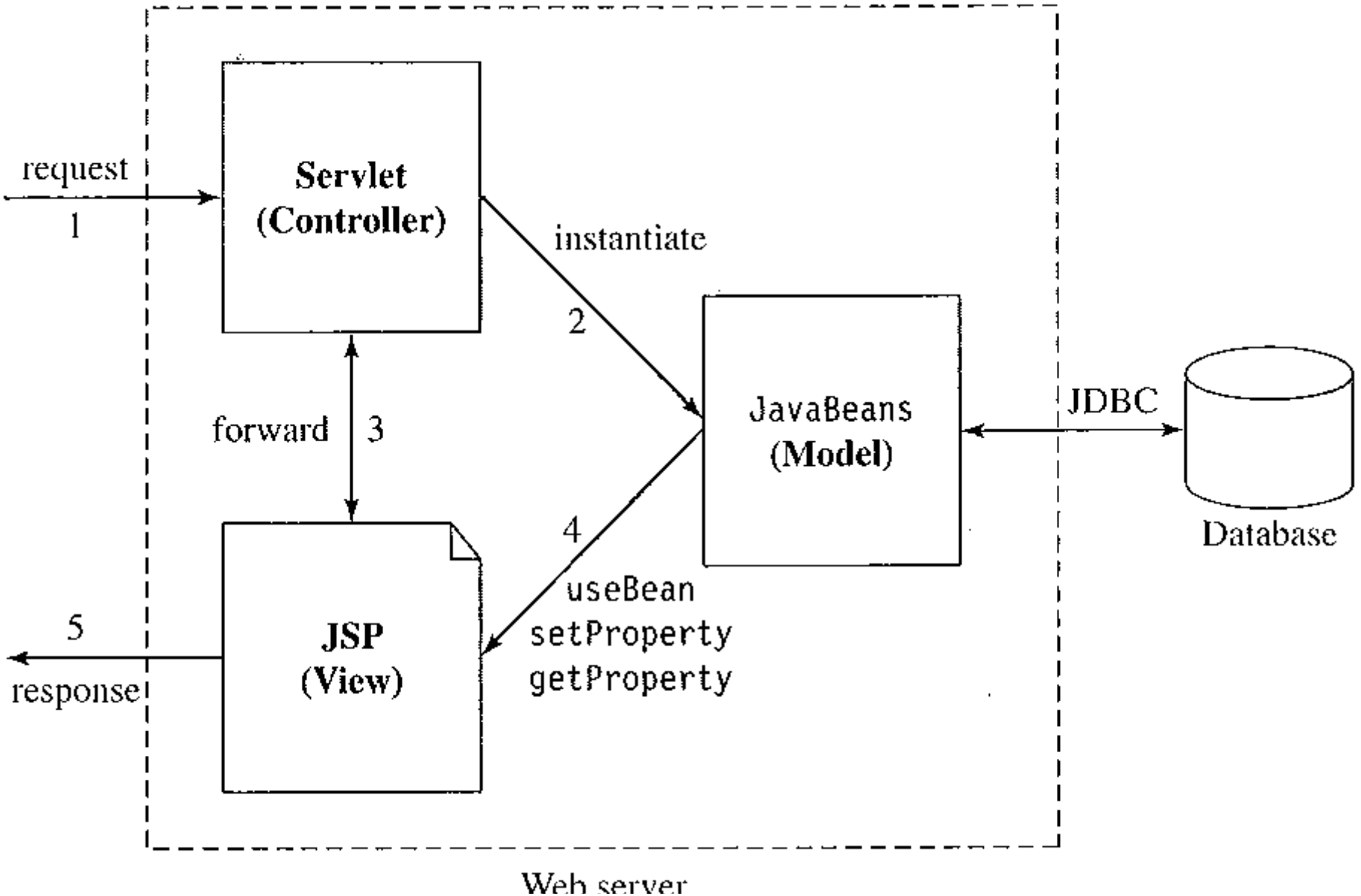


54

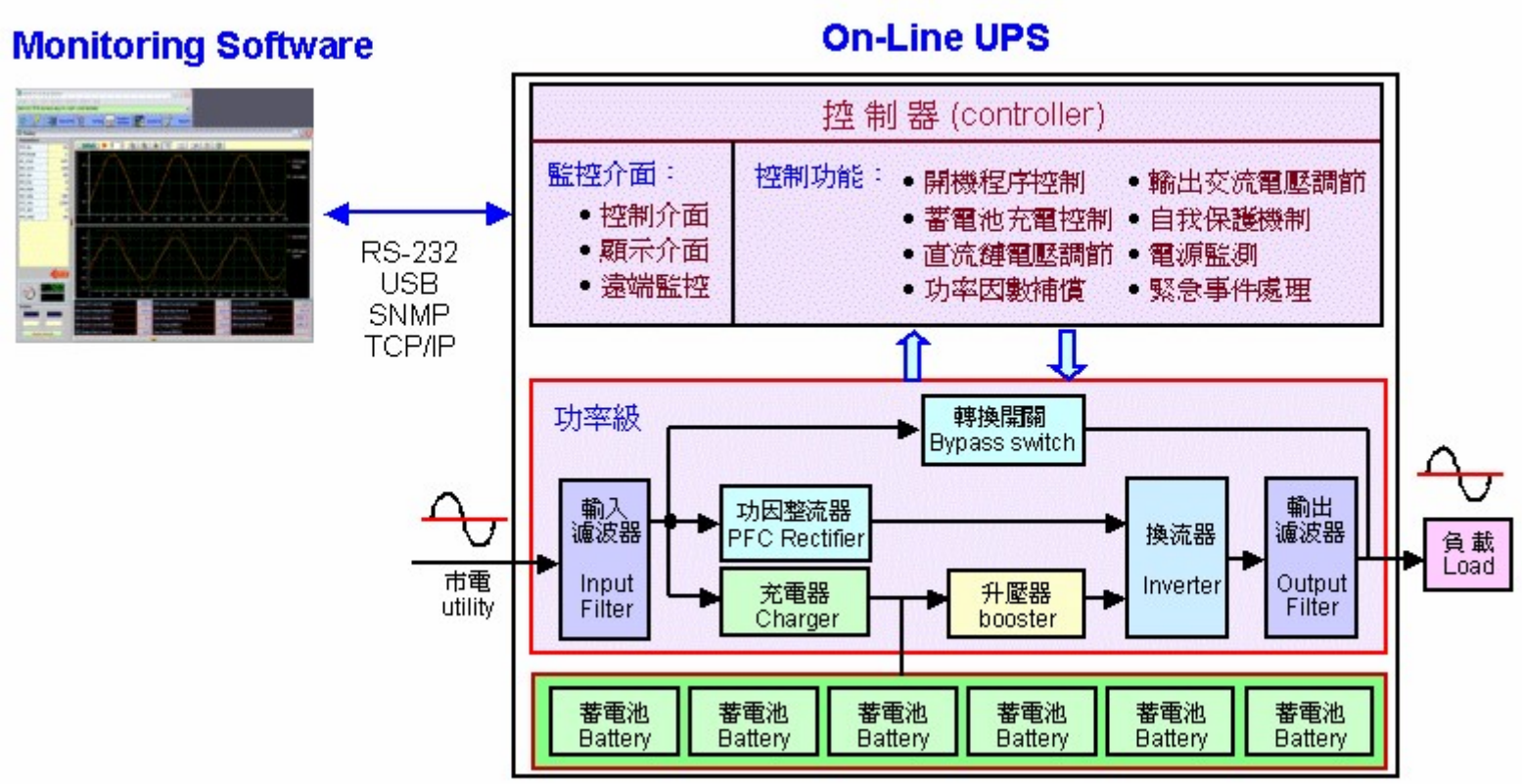
Example: JVM

- Code written in Java is transformed into platform-neutral binary code.
- JVM is platform-specific in that there are different implementations of the JVM for each operating system and processor.
- The Java binary code is delivered to the JVM for interpretation.
- This two-step translation process allows platform-neutral source code and the delivery of binary code, while maintaining platform independency.

55

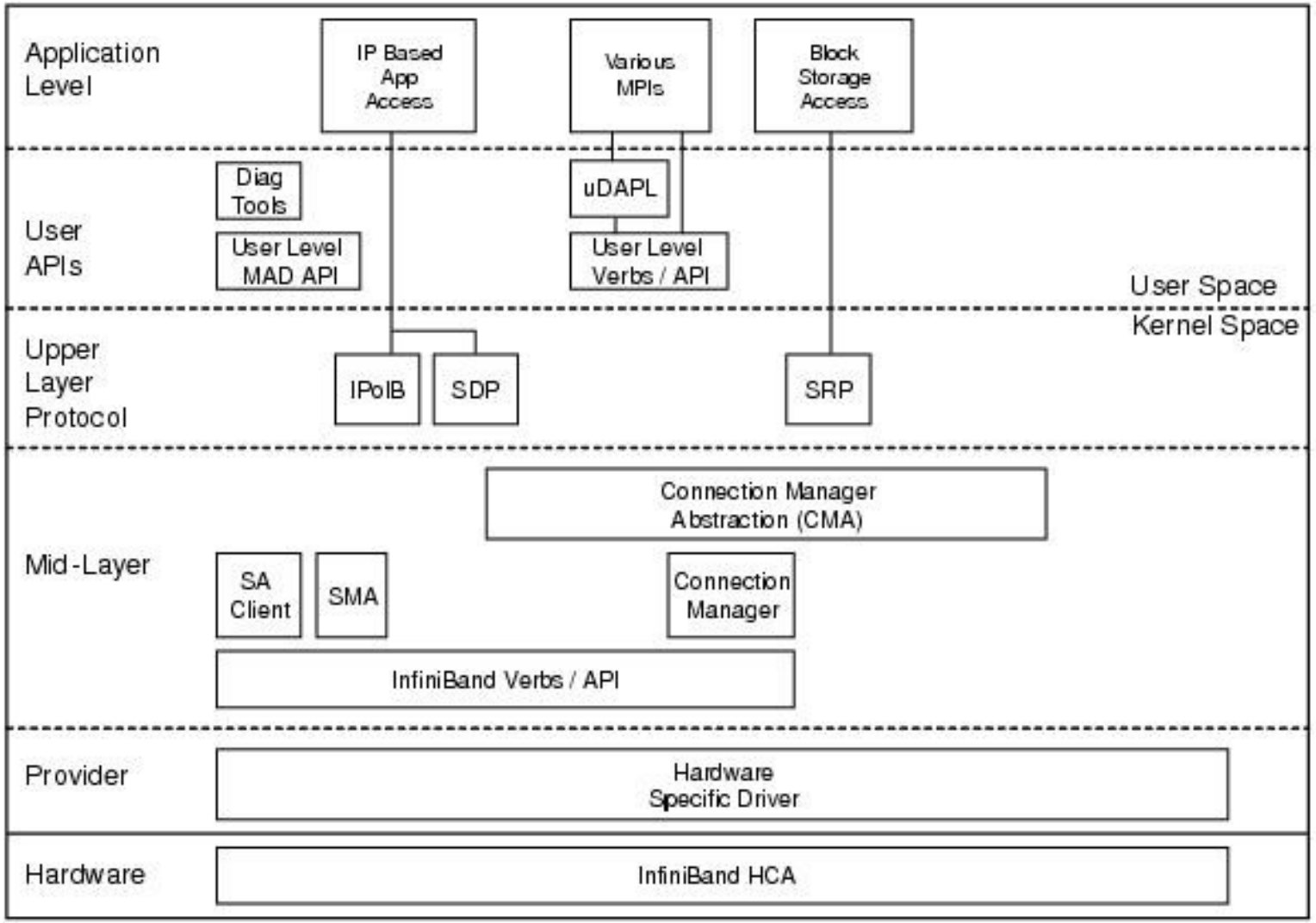

request
1
Servlet
(Controller)
instantiate
2
forward
3
JavaBeans
(Model)
JDBC
Database
4
useBean
setProperty
getProperty
5
response
JSP
(View)
Web server

DSP-Controlled UPS for Smart Power Supplying and Protection



A multiple rate digital controller generates all the PWM control signals for the power stage by using a set of synchronously detected feedback signals.

57



IPoIB	IP over InfiniBand
SDP	Sockets Direct Protocol
SRP	SCSI RDMA Protocol (Initiator)

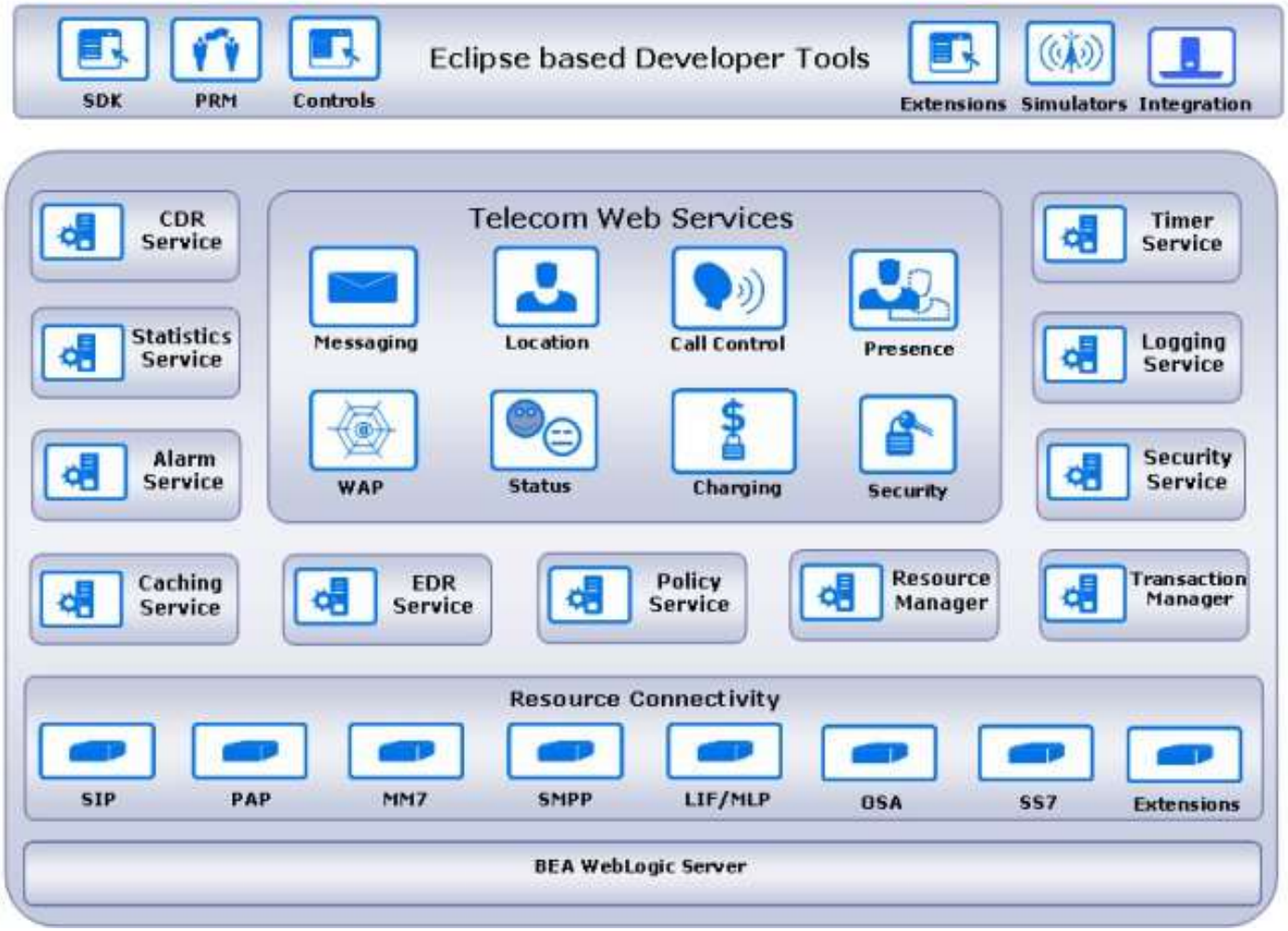
MPI	Message Passing Interface
UDAPL	User Direct Access Programming Lib
SA	Subnet Administrator

MAD	Management Datagram
SMA	Subnet Manager Agent
HCA	Host Channel Adapter

180411

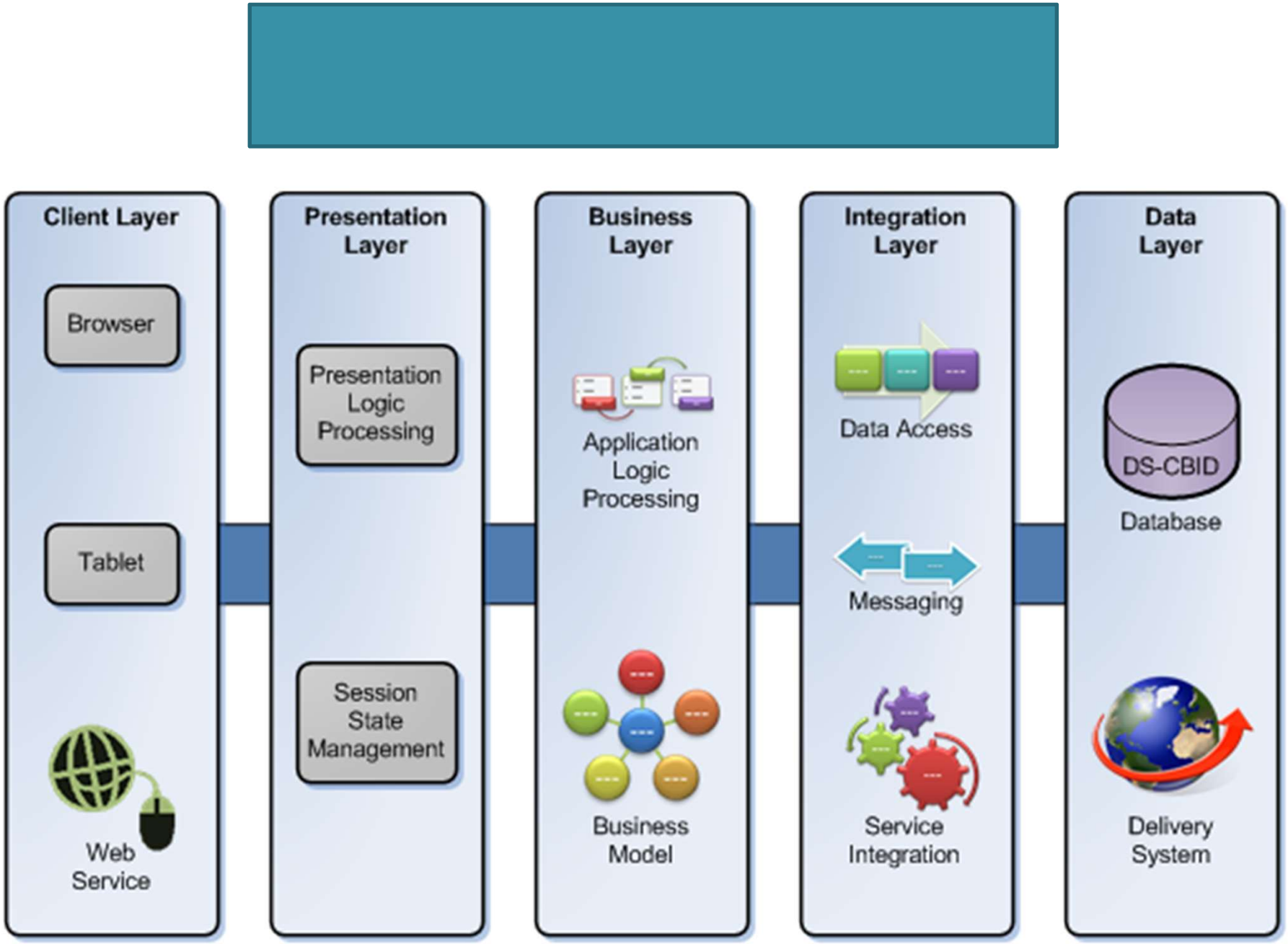
The Cisco IB HCA offers high-performance 10-Gbps InfiniBand connectivity to PCI-X and PCI-Express-based servers.

58

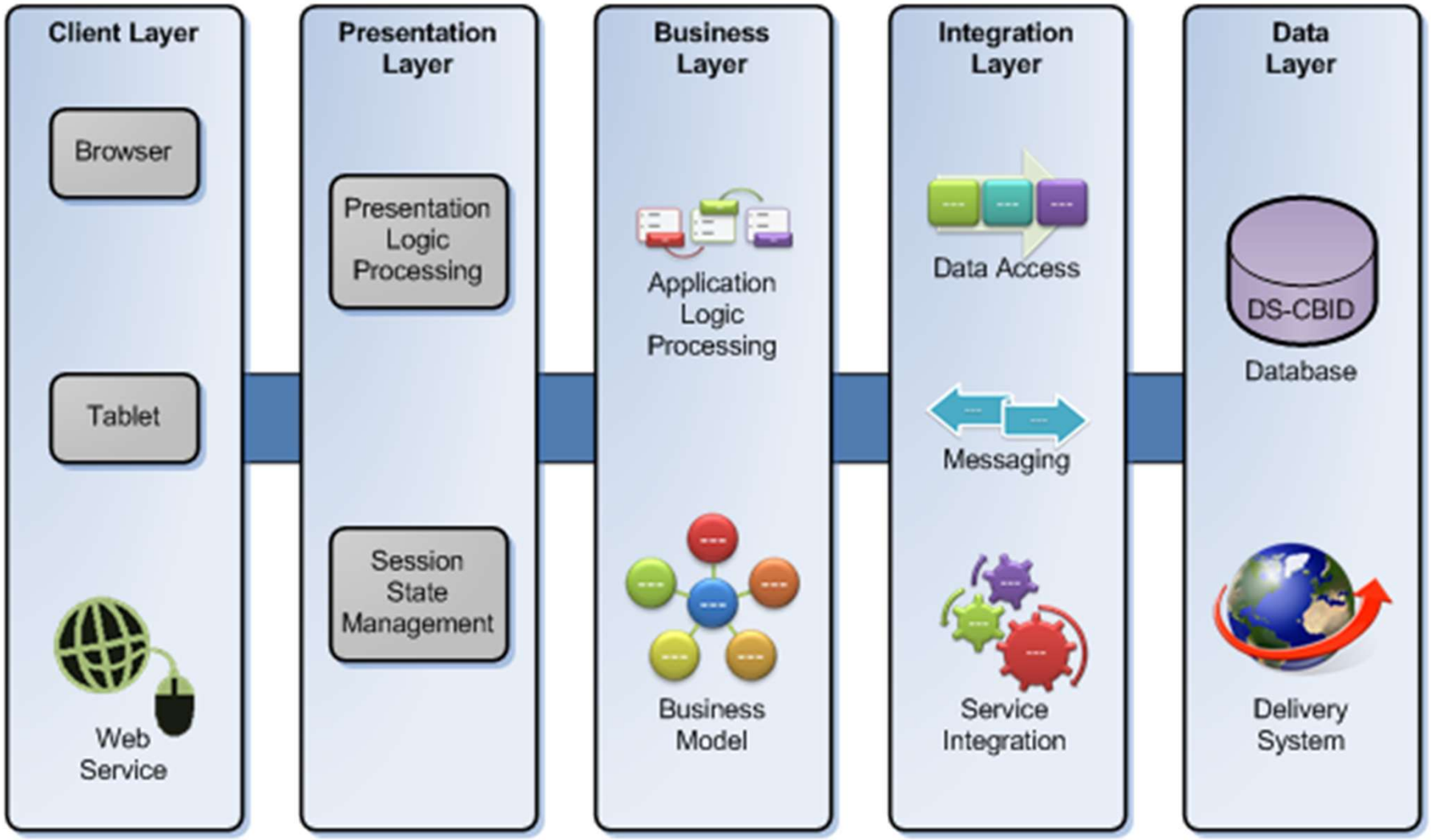


All traffic in Network Gatekeeper flows in traffic paths.
A traffic path consists of an application-facing interface, with Web Services Security enforcement, a Service Capability, and a network plug-in, where requests are translated between the application-facing interface and underlying network node protocols.

59


Client Layer
Browser
Tablet
Web Service
Presentation Layer
Presentation Logic Processing
Session State Management
Business Layer
Application Logic Processing
Business Model
Integration Layer
Data Access
Messaging
Service Integration
Data Layer
DS-CBID
Database
Delivery System

n-Tier Architecture



References

- Software architecture in practice - second edition
  Len Bass, Paul Clements, Rick Kazman
  Addison Wesley, 2003

- Pattern-oriented software architecture
  Buschmann, Meunier, Rohnert, Sommerlad, Stal
  Wiley, 1996

- Applied software architecture
  Hofmeister, Nord, Soni
  Addison Wesley, 2000

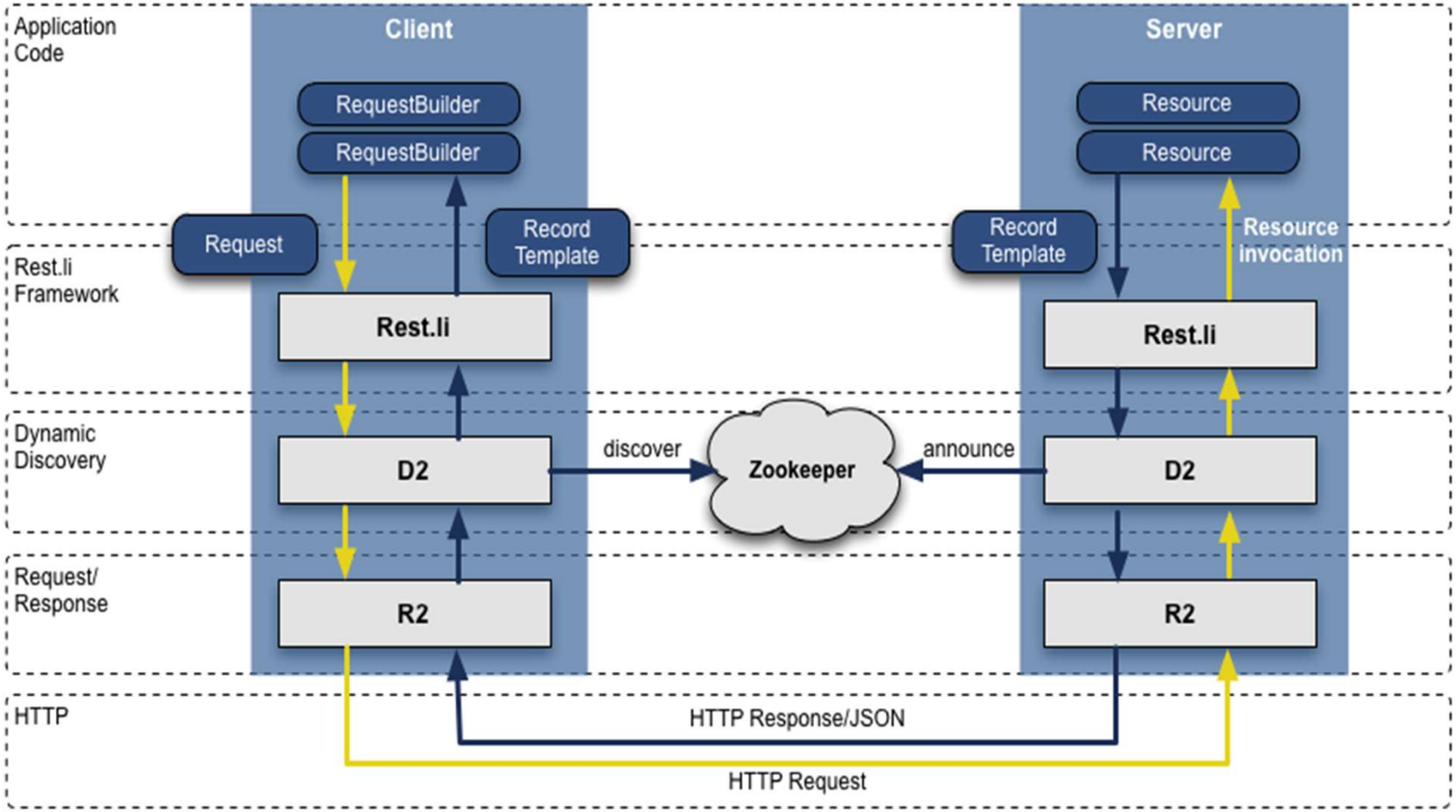
- Design and use of software architectures
  Jan Bosch
  Addison Wesley, 2000

62

For the final evaluation

- You should know
  - Challenges and issues of software architecture
- You should be able to
  - Read/complete an architectural description
  - Recognize an architectural style (among those which were presented)

Exercise



Reformulate the previous schema into an architectural description that follows the principles of (1) separation of concerns and (2) abstraction.