Software architecture

Representation

Gabriel Pedraza

pedraza@imag.fr

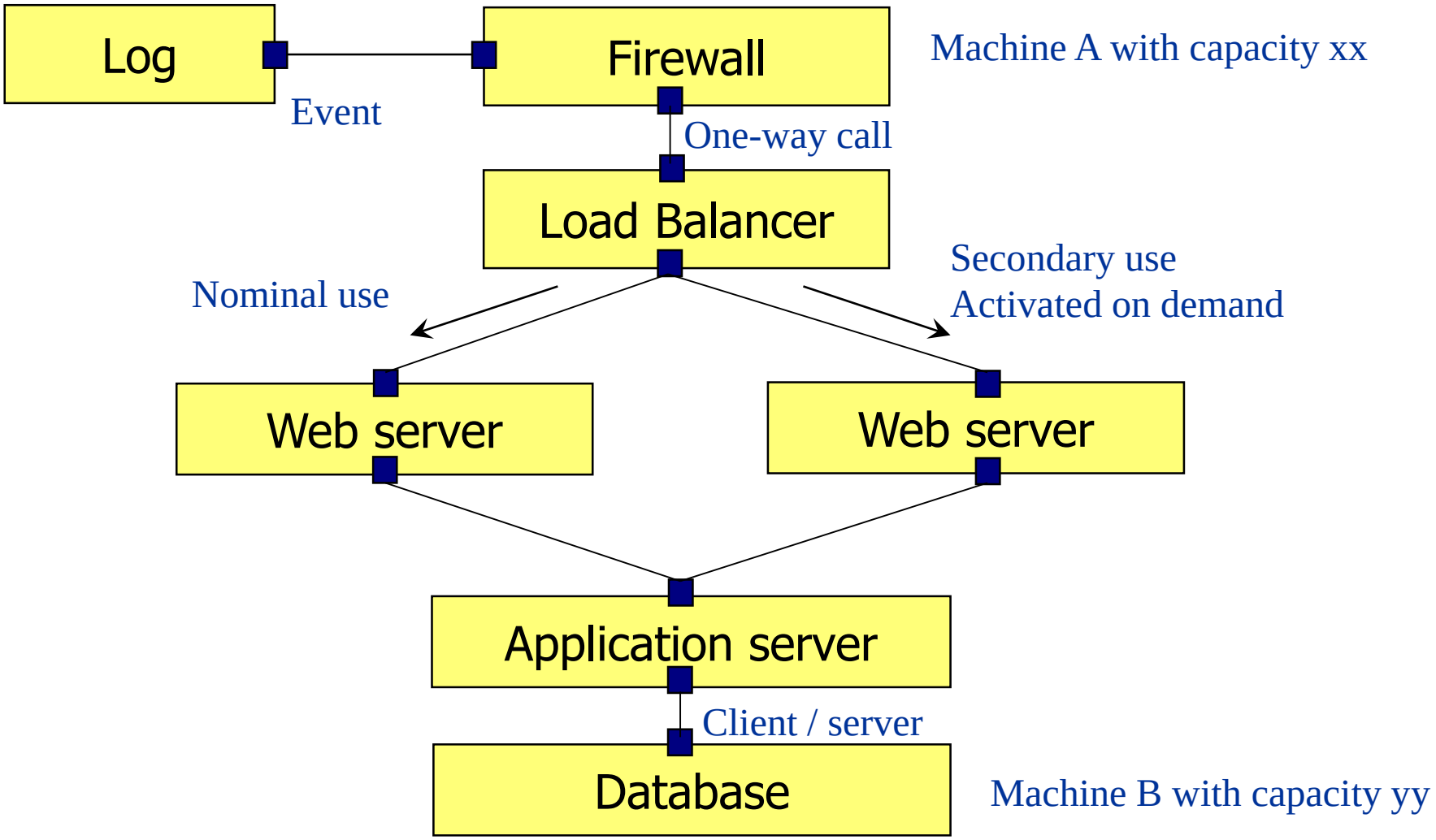
Outline

- Architectural views
- Logical views
- Dynamic views
- Allocation views
- Conclusion

Representing architectures

- The goal is to represent all the information related to software components
  - Structure and interfaces
  - Interaction
  - Properties and constraints
  - Runtime support, …
  - Partnerships
- Ideally, this information should be represented on a single schema

Example


Log
Firewall
Machine A with capacity xx
Event
One-way call
Load Balancer
Secondary use
Activated on demand
Nominal use
Web server
Web server
Application server
Client / server
Database
Machine B with capacity yy

Verdict

- Heterogeneous, incomplete, ambiguous
  - Very different pieces of information targeting various people
    - Concerns mixing
    - Hard to read
  - Lots of information are missing
    - How many machines?
    - Connectors between web servers and application server?
    - Protocol between firewall and load balancer?
- Need of structured, repeatable approach

Analogy

- In buildings, several structures coexist
  - Topology
  - Electrical wires
  - Plumbery
  - HVAC structures

- Specialized plans used as specifications
  - Used by different persons
    - Maçons, plumbers, …
  - Used to specify different properties
    - Density, diameters, pressure, …

Application to software

- Several structures also coexist in software programs (Parnas, 74)
  - Different programming tools are provided to express these structures
    - Type definition, loop operators, …
- This has to be applied to software architecture

Application to software

- Several structures also coexist in software programs (Parnas, 74)
  - Different programming tools are provided to express these structures
    - Type definition, loop operators, …
- This has to be applied to software architecture

Software views - definition

- A view provides a focused perspective on a software
  - Separation of concern

- A view defines
  - Soft. elements that can be represented on the view
  - Relations that can be represented
  - Constraints that can be represented
  - A formalism and a vocabulary

Usual views

- Logical views
  - How the software is structured into components
- Dynamic views
  - How the software behaves
- Allocation views
  - How the components are executed

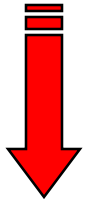
Lost?

- Not having a single view brings good and bad points
  - Better coherence
  - Complexity reduction
  - Focus improvement

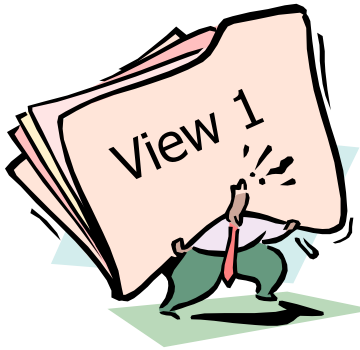
  - The architecture dematerializes …
  - Relationships between views …

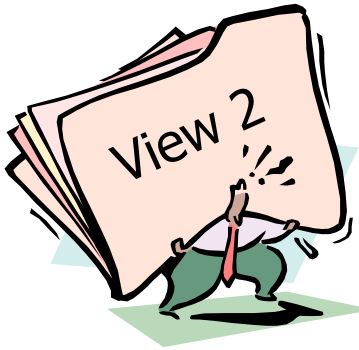
Synthesis

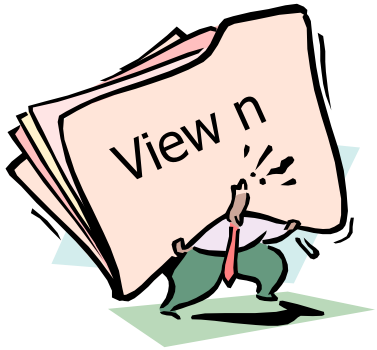
Architecture = component + connector



Representation = a set of views







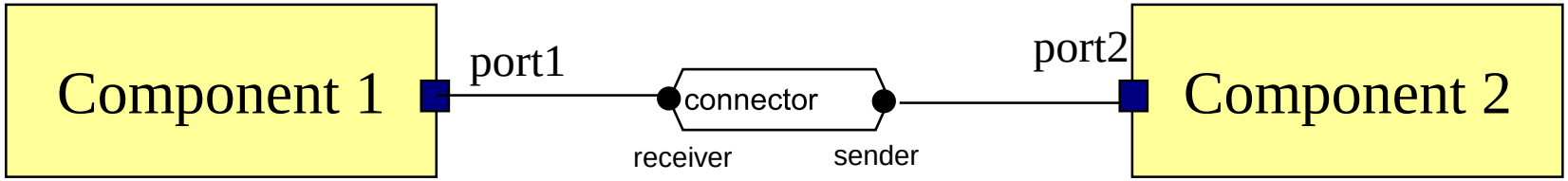
Outline

- Architectural views
- Logical views
- Dynamic views
- Allocation views
- Conclusion

Logical view

- It defines the structure of the architecture
  - Decomposition into logical elements

- All components and connectors are described
  - Connections are potential
  - Connectors can be complex



Logical view: components

- A component is defined by
  - A unique name
  - Several ports
  - Properties (performance, persistence, robustness, …)
- A port contains
  - Provided services (functional, lifecycle related, …)
  - Required services
  - Constraints (pre and post, scheduling, …)

Logical view: interactions

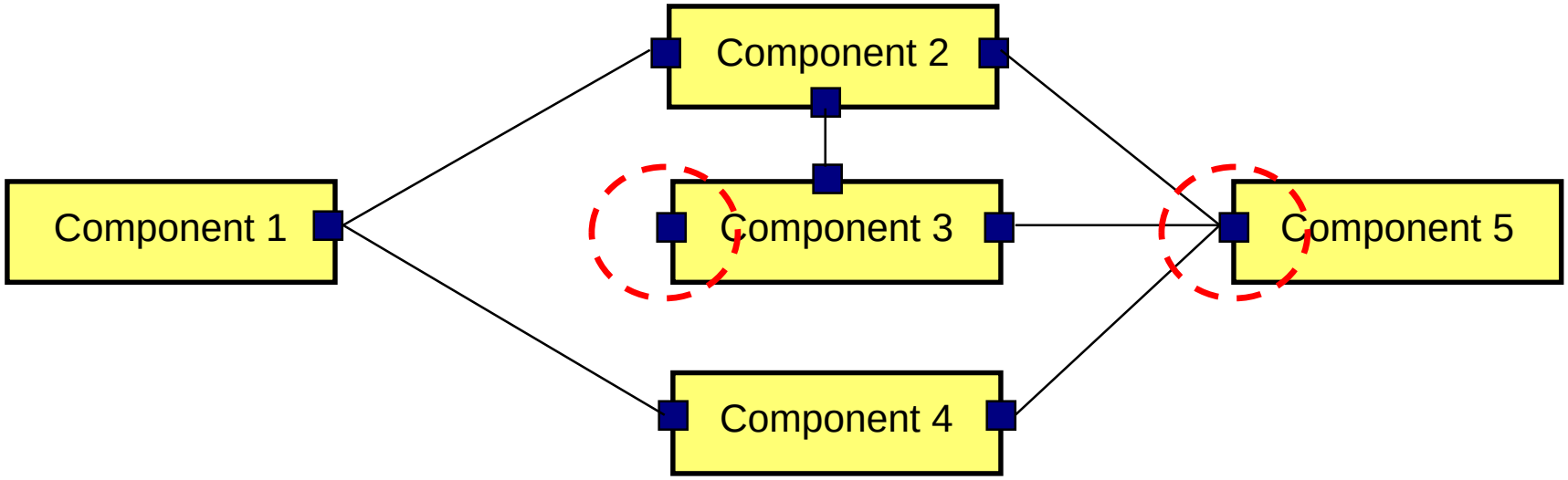
- Role
  - Communication
  - Coordination
  - Conversion
  - Facilitation
- Communication modes
  - « Procedure Call » (PC, RPC, C/S, methods, …)
  - « event » (P/S, broadcast)
  - « Stream », …

Logical view: interest

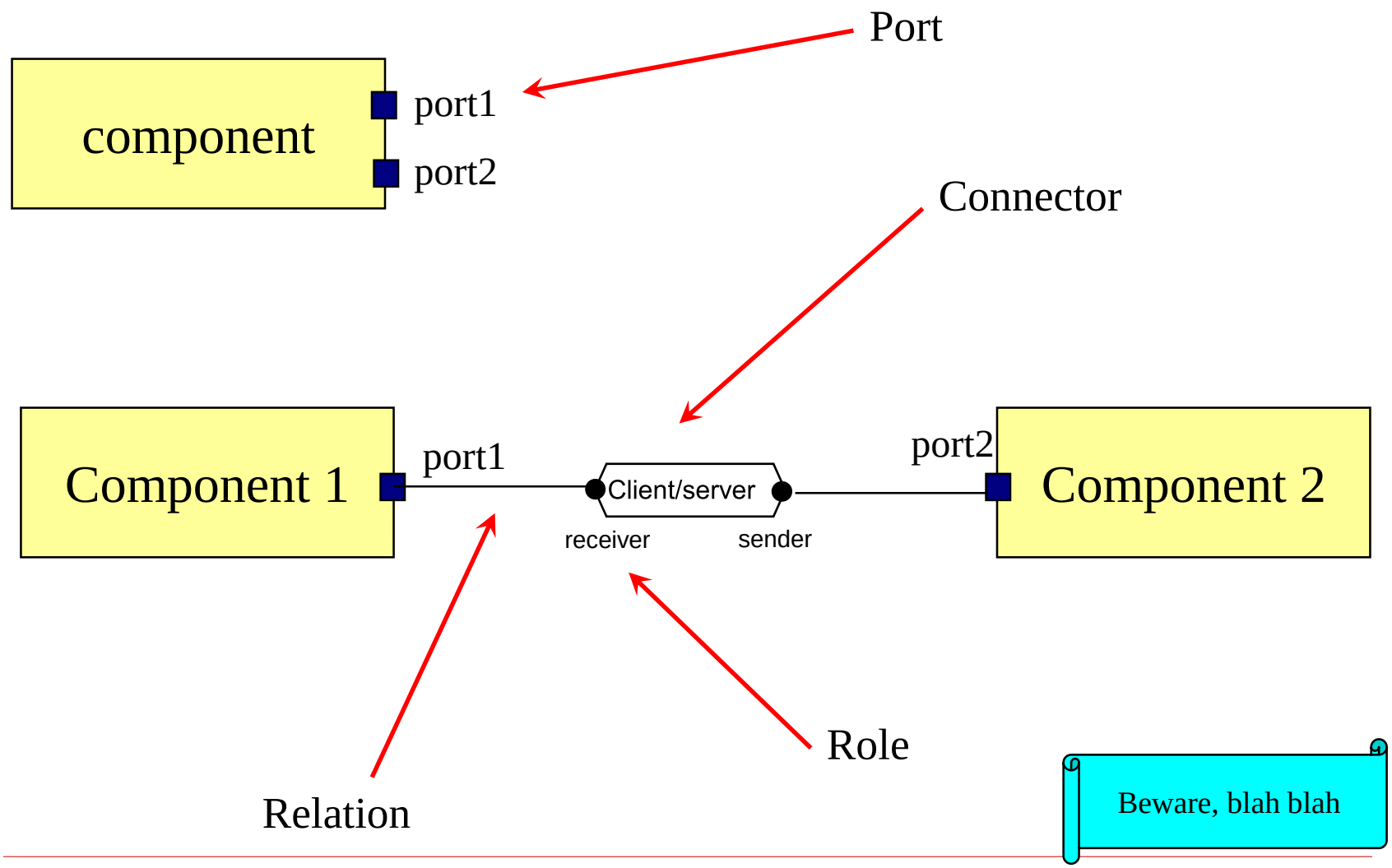
- Provides answers to the following questions
  - What are the main computing elements?
  - What are their relations?
  - What are the main data sources?
  - What are the need in terms of supporting runtime?
  - What are the possible bottlenecks?
  - What are the critical paths?
  - What are the levels of coupling and cohesion?

Logical view: usage example

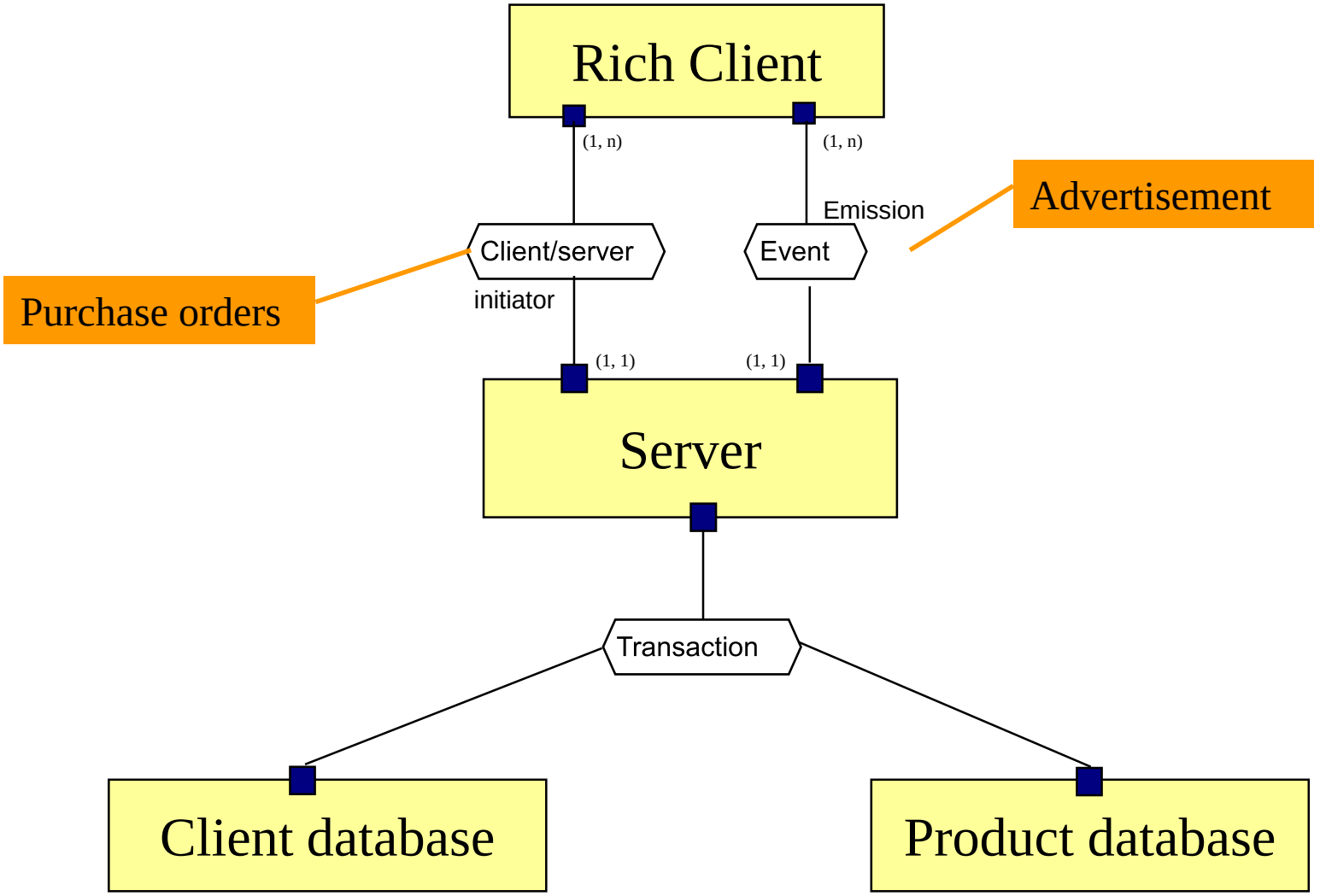
- Issues
  - Bottlenecks,
  - Use of every interface



Logical view: formalism



Example: a vending application

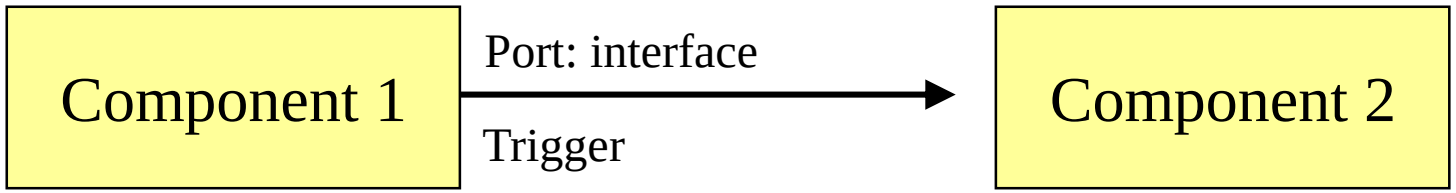


Outline

- Architectural views
- Logical views
- Dynamic views
- Allocation views
- Conclusion

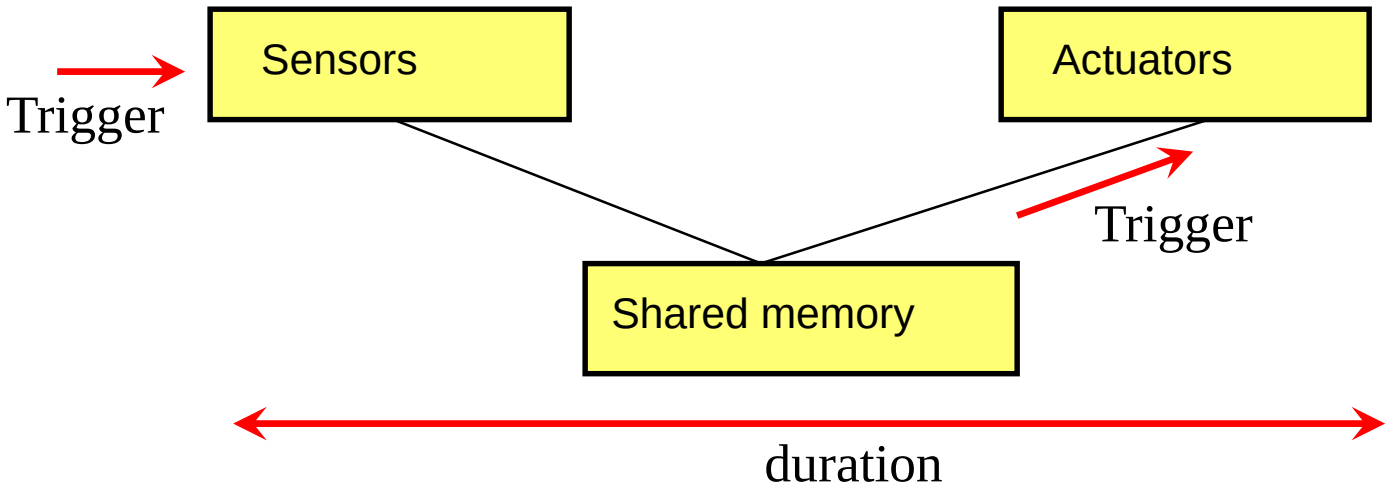
Dynamic view

- It defines interaction within the architecture
  - When and why do they occur?
  - How are they realized?
- Only effective relations are presented



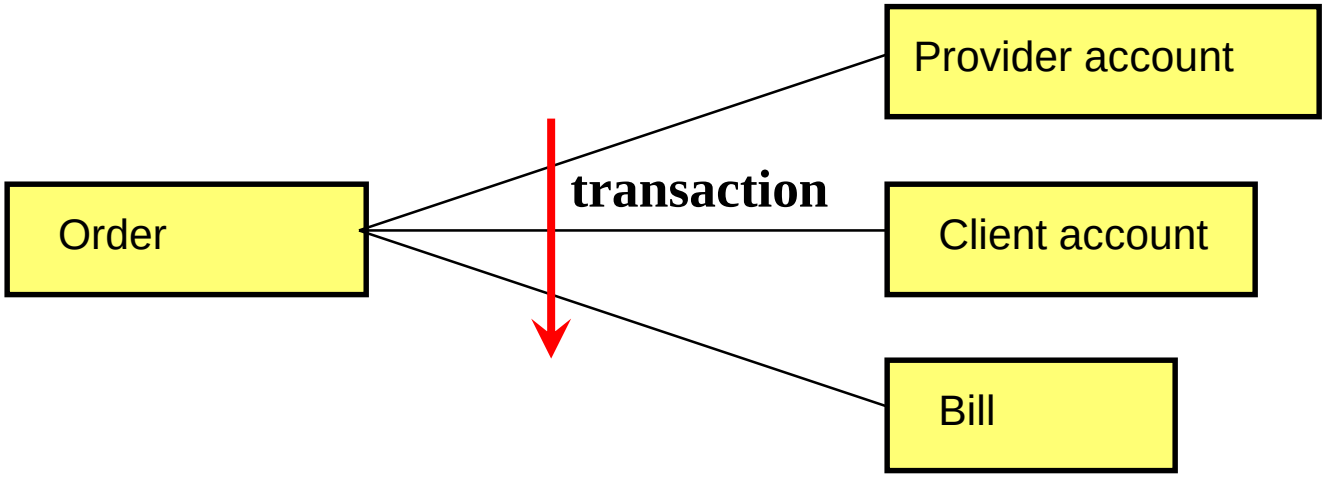
Dynamic view: usage example (1)

- Real time systems
  - Events and durations are specified to ensure performance, reliability



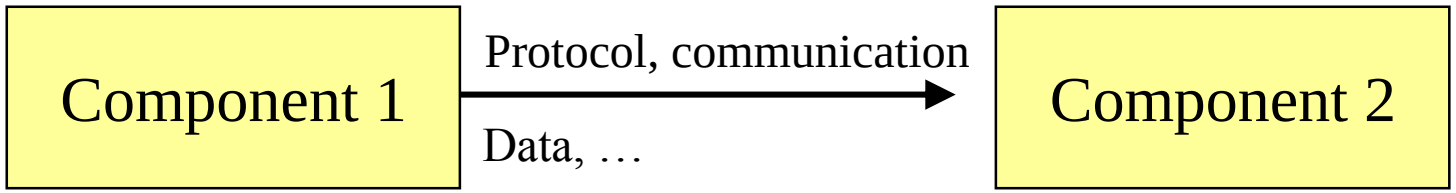
Dynamic view: usage example (2)

- Banking systems
  - Transactions are specified to allow rollbacks



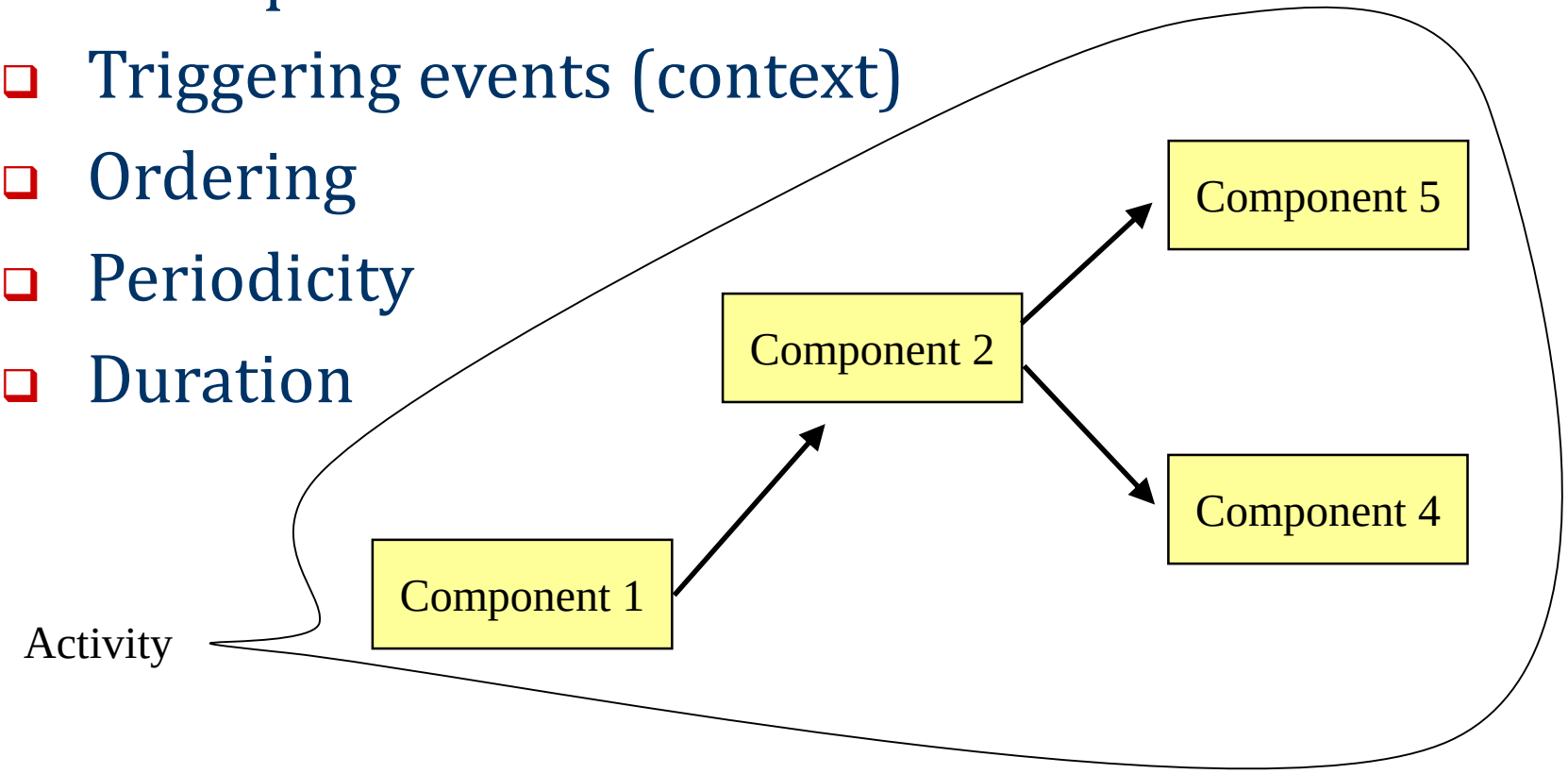
Dynamic view: interactions

- An interaction is defined by
  - Communication type
  - Exchanged data
  - Concurrency (possible)



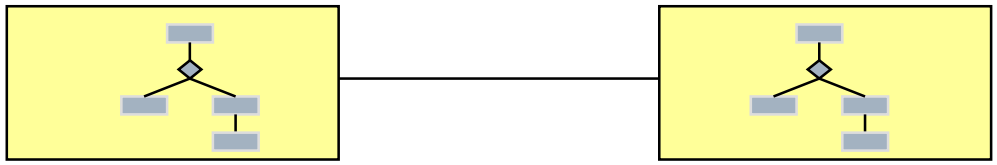
Dynamic view: activities

- An activity is defined by
  - A unique name
  - Triggering events (context)
  - Ordering
  - Periodicity
  - Duration



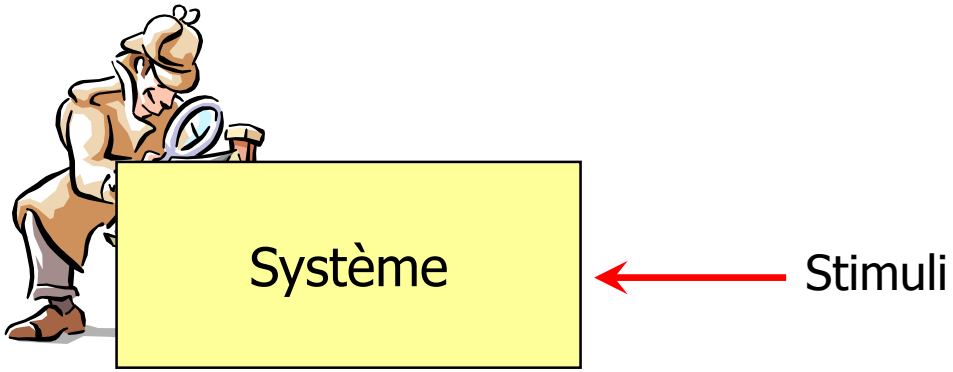
Representing dynamics

- Trace-based approach
  - Explicit description of interactions in a system after a given stimuli
  - Empirical approach
- Model-based approach
  - Use of a mathematical language to model system dynamics
  - Formal approach



Traces

- Principles
  - Trace all the interactions between structural elements during the execution of a scenario
  - Traces are complete for a given scenario
  - Union of all possible scenarios provides a complete behavioral model (not feasible)
- Popular approach
  - UML
  - Architecture

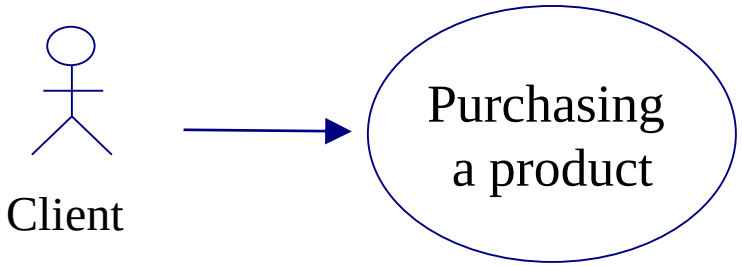


Scenario

- A scenario represents a system usage
  - A scenario is triggered by an event in a context
  - It triggers a sequence of interactions between components
  - It exemplifies an internal behavior of the system

- It is not a use case

Scenario - representation



Scenario : nominal purchase

Description

The scenario begins when a client pushes « start » button.

Then , the client selects a product et introduces his card.

The corresponding account is debited and the product is provided.

End of the scenario.

Actors

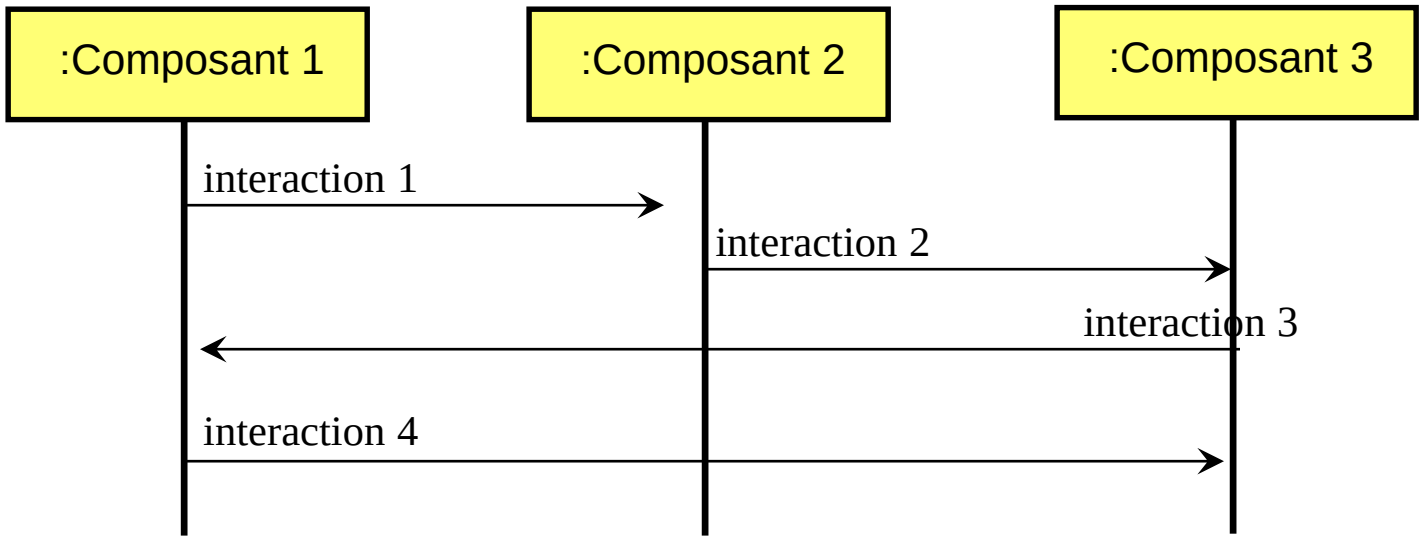
1. The client

Exceptions

- The client can leave before entering his card
- Etc.

Scenario – advanced representation

- Sequence diagrams



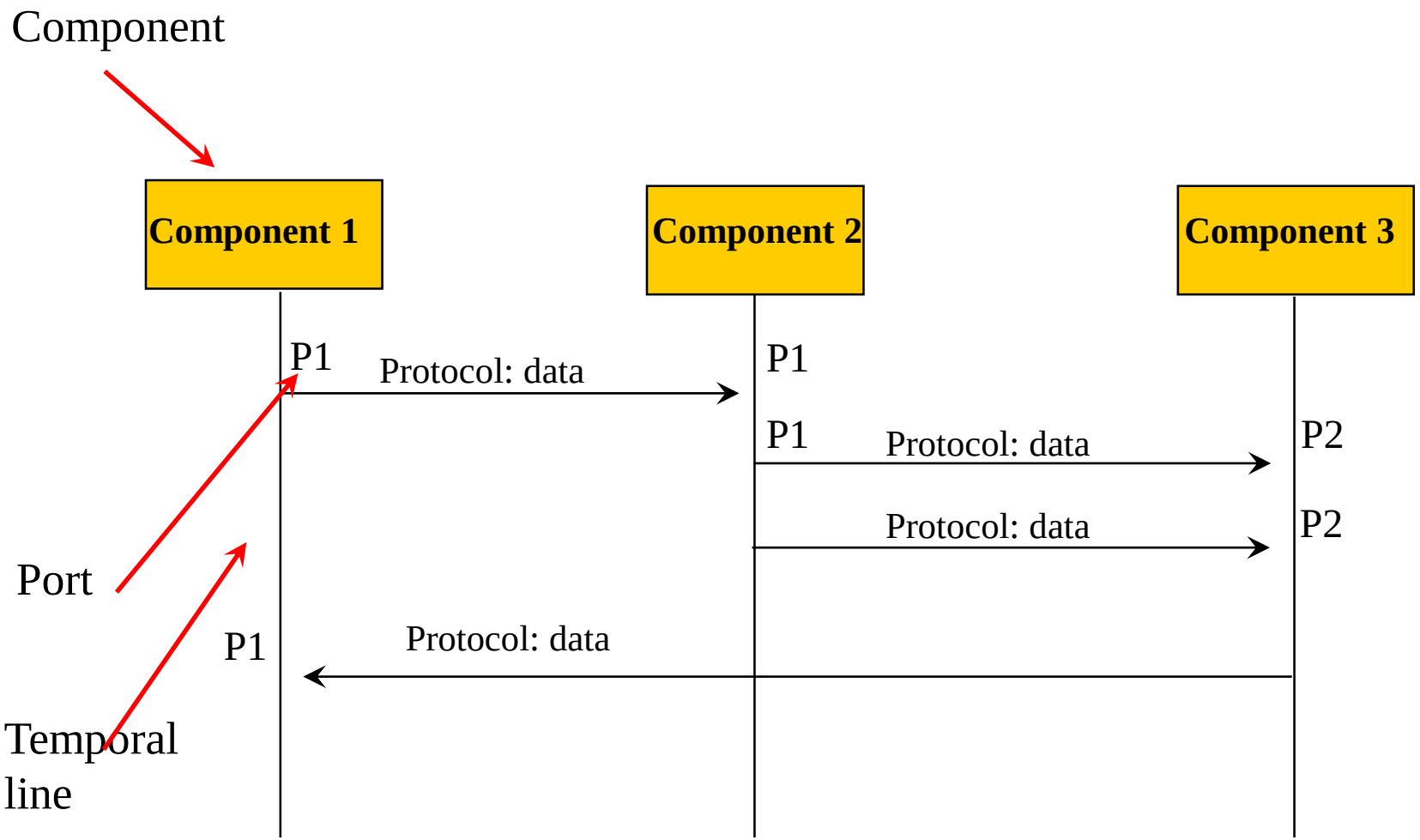
Side notes

- These diagrams only contain elements implied in a scenario
- Invoked interfaces are on the arrows
  - more or less detailed
  - depends on the design advancement
- Concurrent activities are hard to express
- A constraint language can be used to get more precise
  - OCL (Object Constraint Language) of UML is an example

Interests of traces

- Advantages
  - express (partially) a system dynamics
  - provide a good support to design
  - Pragmatic and easy to use
  - Allow a few validations to be performed
    - Paths complexity, bottlenecks, …
- Problems
  - cannot cover the whole dynamics
  - Very difficult to identify the good set of traces
  - Analysis/validation remains limited

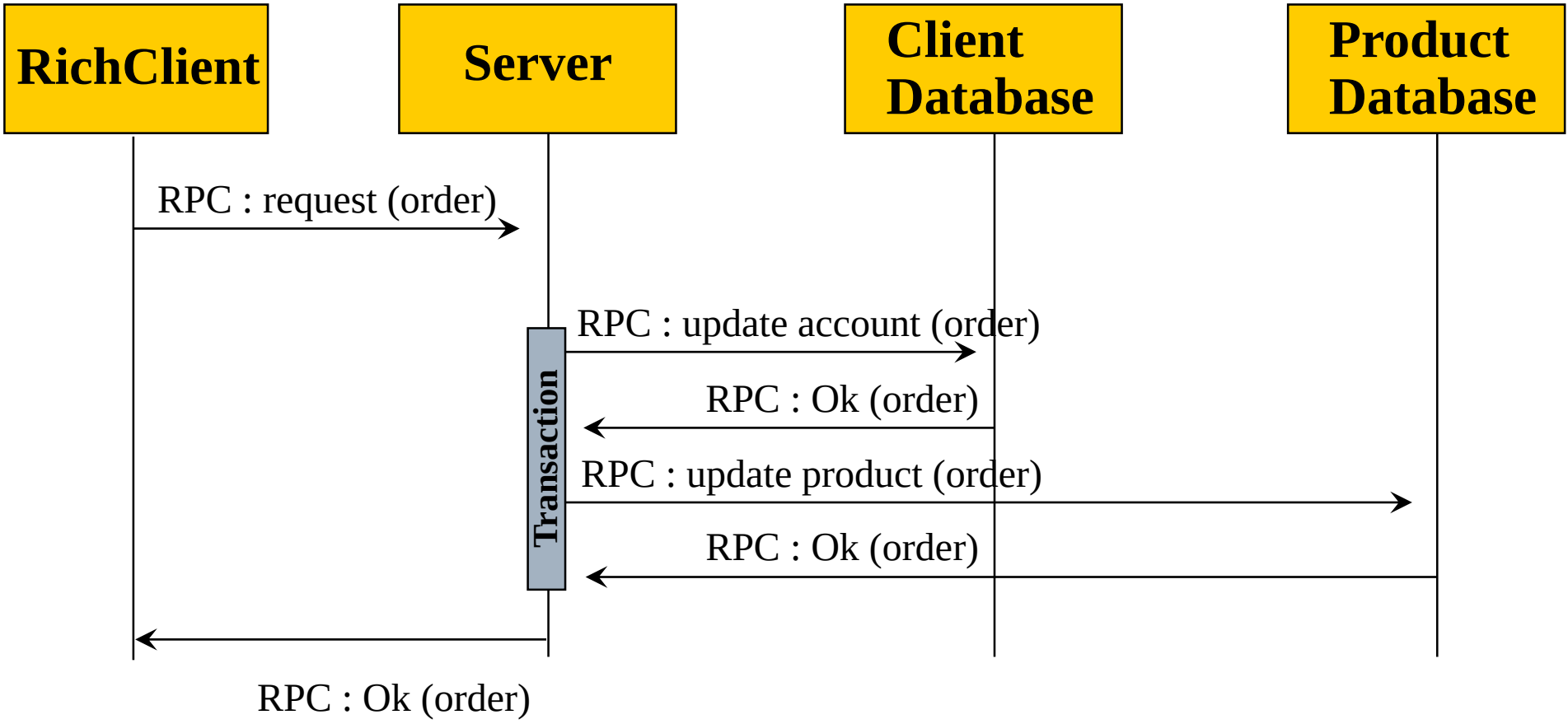
Formalism


Component
Component 1
Component 2
Component 3
P1
Protocol: data
P1
P1
Protocol: data
P2
Protocol: data
P2
Port
P1
Protocol: data
Temporal line

Example

- Scenario
  - Description: Shows what happens when a borne sends a command to the main server
  - Triggering event: a payment is done. Such an event may happen anytime
  - Context: nominal behavior. The main server is not overloaded and there is no concern with the client credit card.

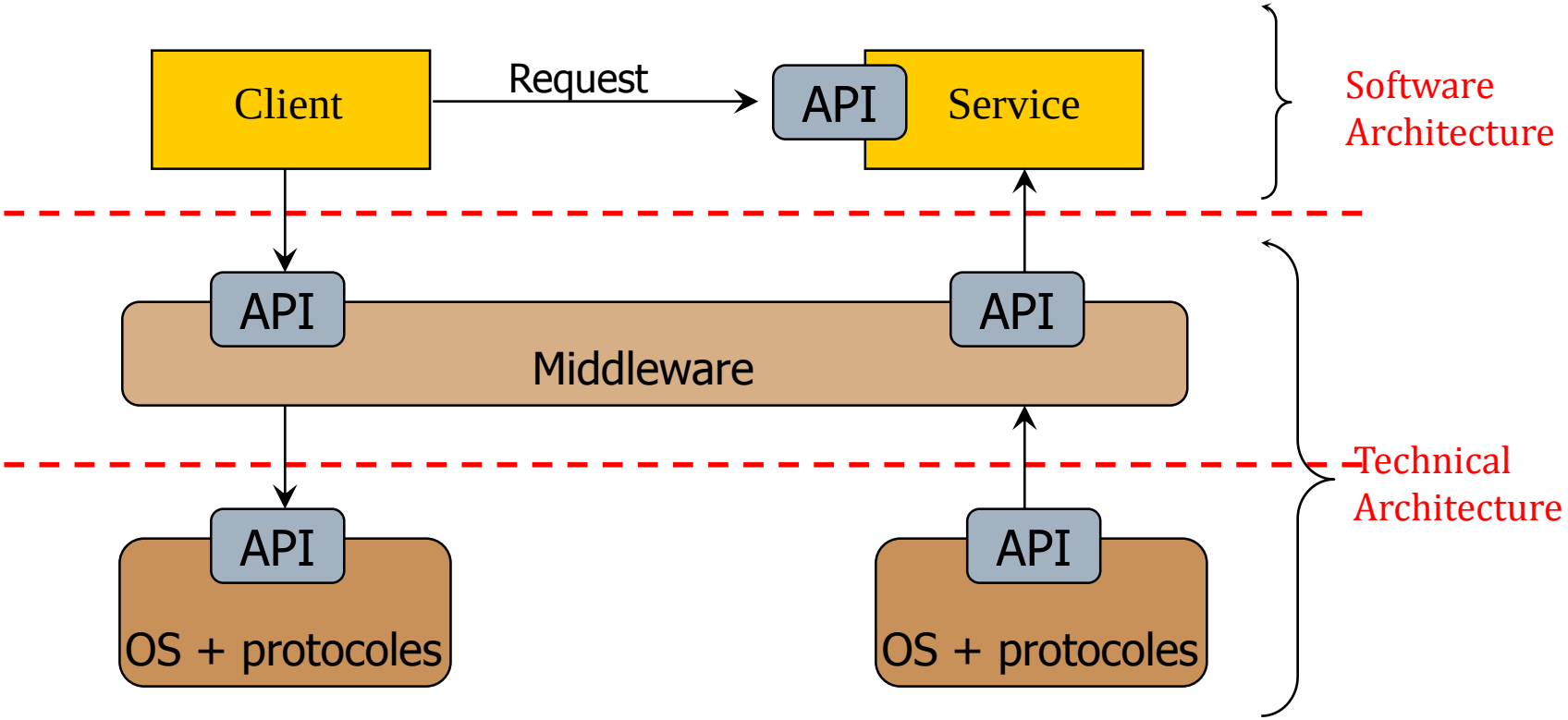
Example


RichClient
Server
Client Database
Product Database
RPC : request (order)
RPC : update account (order)
Transaction
RPC : Ok (order)
RPC : update product (order)
RPC : Ok (order)
RPC : Ok (order)

Outline

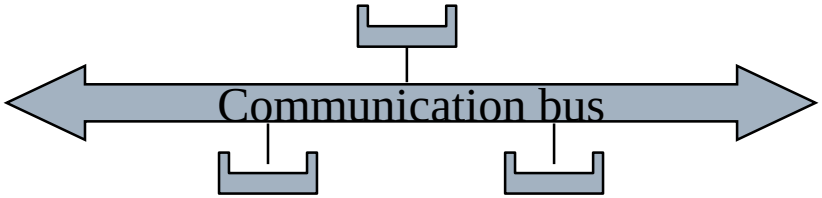
- Architectural views
- Logical views
- Dynamic views
- Allocation views
- Conclusion

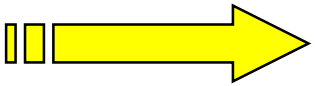
Allocation view (reminder)

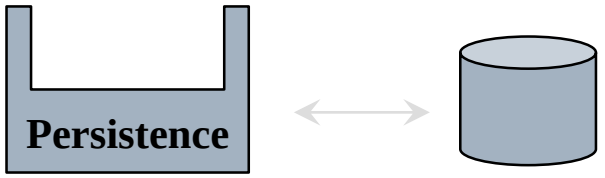


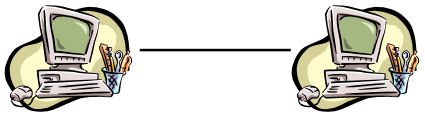
Allocation view

- Represent the mapping of the structural elements onto
  - Physical resources
  - Middleware

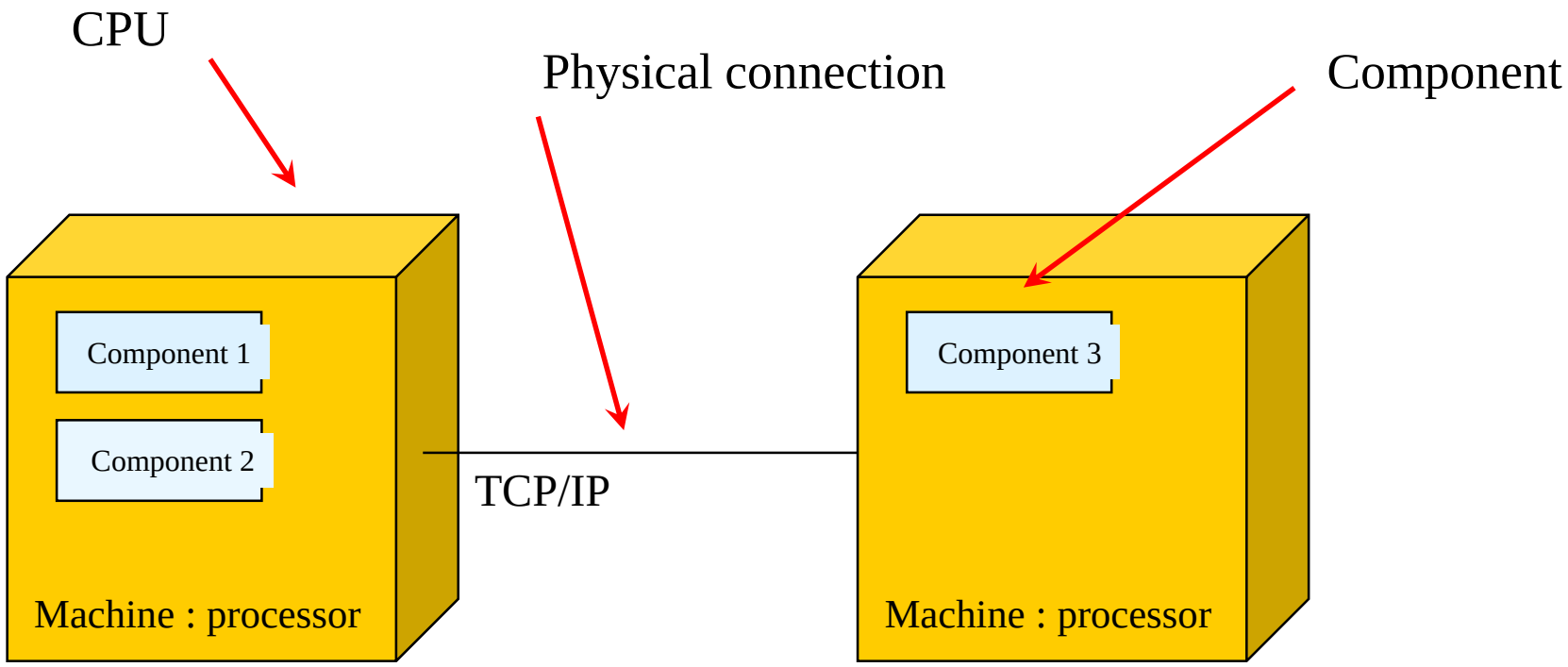




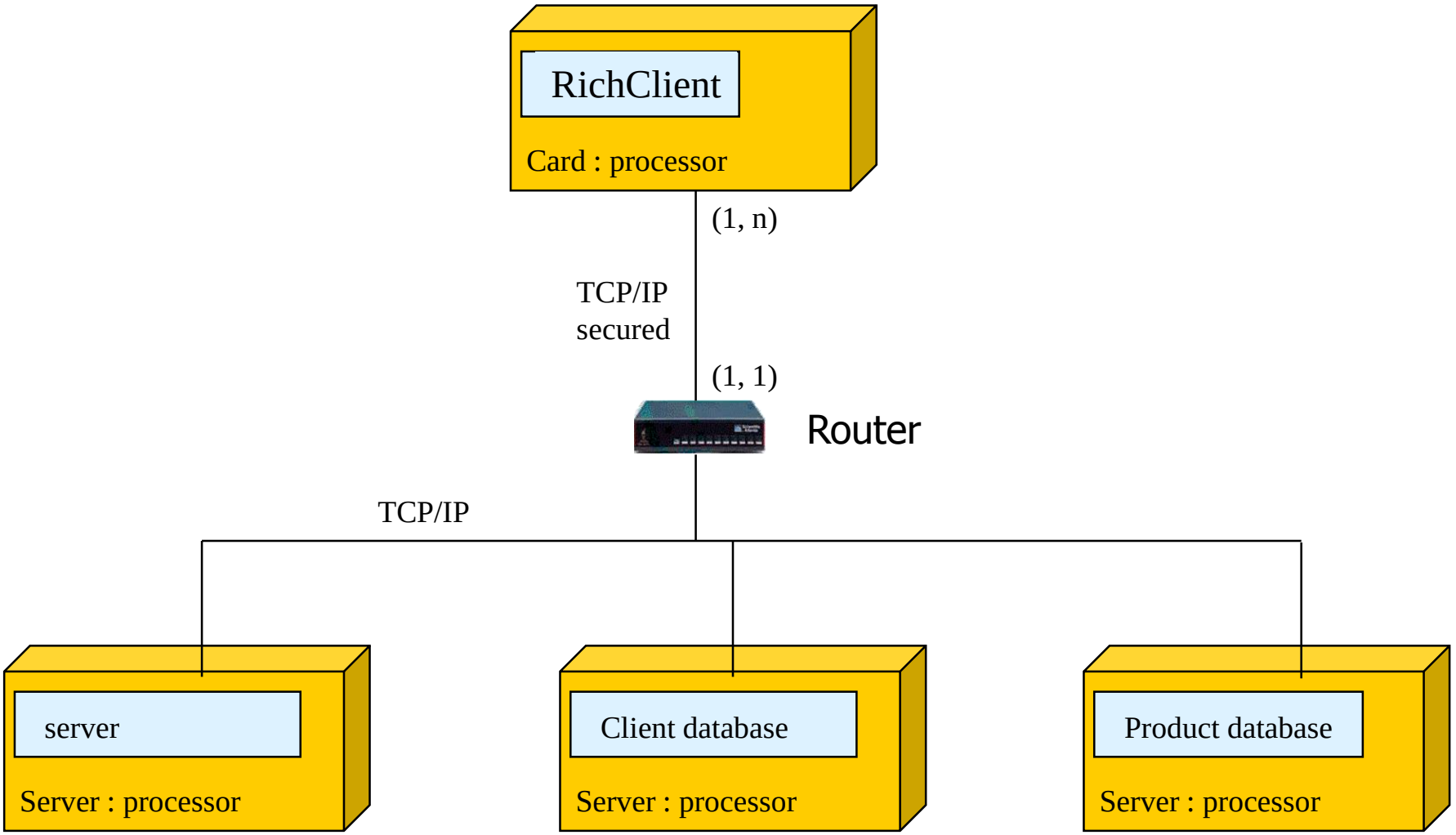




Formalism


CPU
Physical connection
Component
Component 1
Component 2
Component 3
TCP/IP
Machine : processor
Machine : processor

Example


RichClient
Card : processor
(1, n)
TCP/IP
secured
(1, 1)
Router
TCP/IP
server
Server : processor
Client database
Server : processor
Product database
Server : processor

Outline

- Architectural views
- Logical views
- Dynamic views
- Allocation views
- Conclusion

Major points

- An abstract specification
  - Architecture is a model
- Conceptual *vs.* technical detail
  - Different levels of specification
- Architectural views