Architectural **Viewpoints**

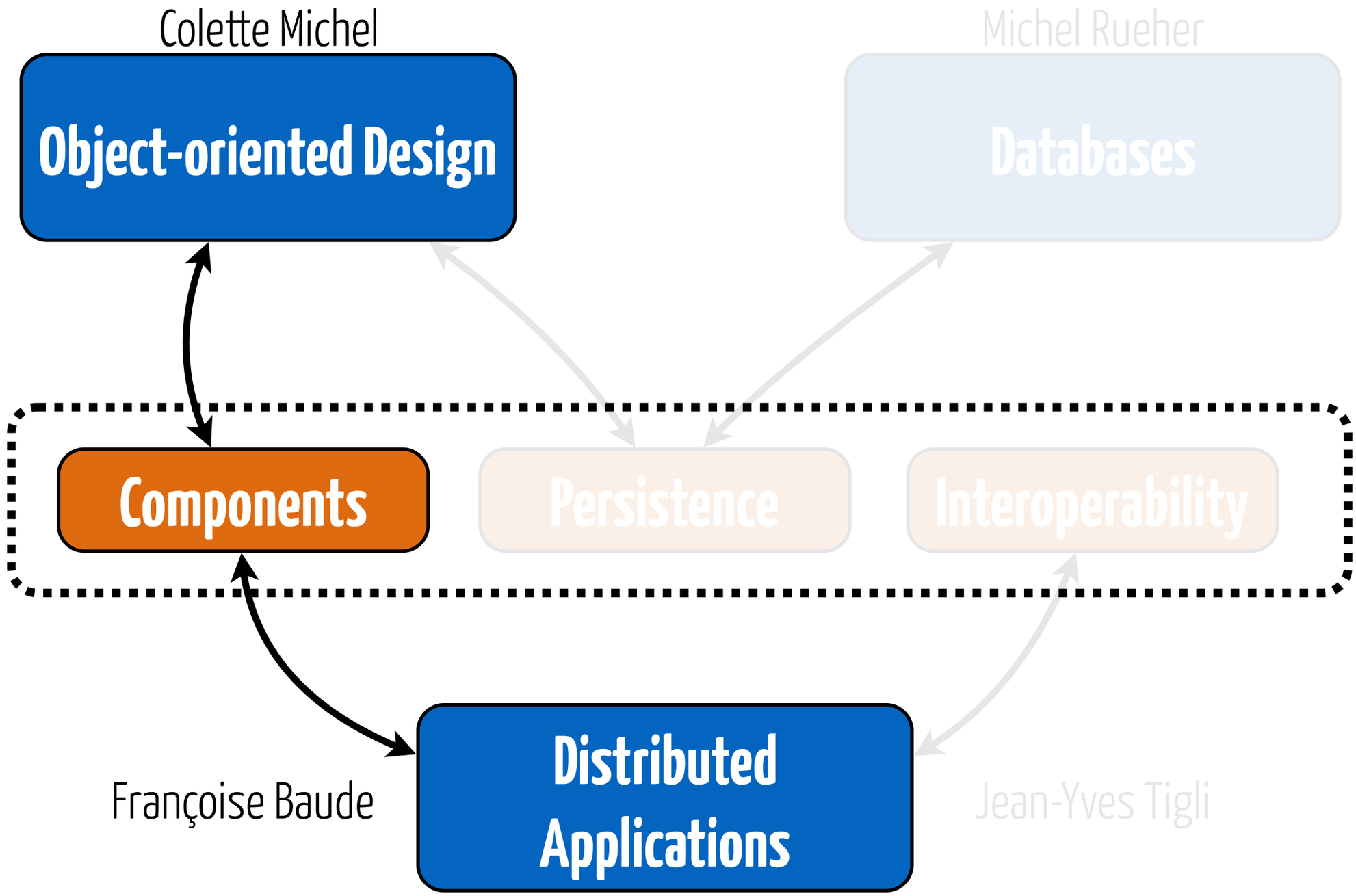
Sébastien Mosser

Lecture #1.3, 09.02.2018

pictures: sxc.hu

CC BY NC

Applications Server: Dependencies



“A **complex system** is much more **effectively** described by **a set of interrelated views** […] than by **a single overloaded model**

[SSA]

“A **view** is a representation of one or more **structural aspect** of an **architecture**

[SSA]

> A **viewpoint** is a **collection** of **patterns**, **templates**, and **conventions** for **constructing one type of view**

[SSA]

Viewpoints and views: Pros

Separation of concerns

Improved developer focus

Communication with stakeholders

Management of complexity

Viewpoints and views: Pitfalls

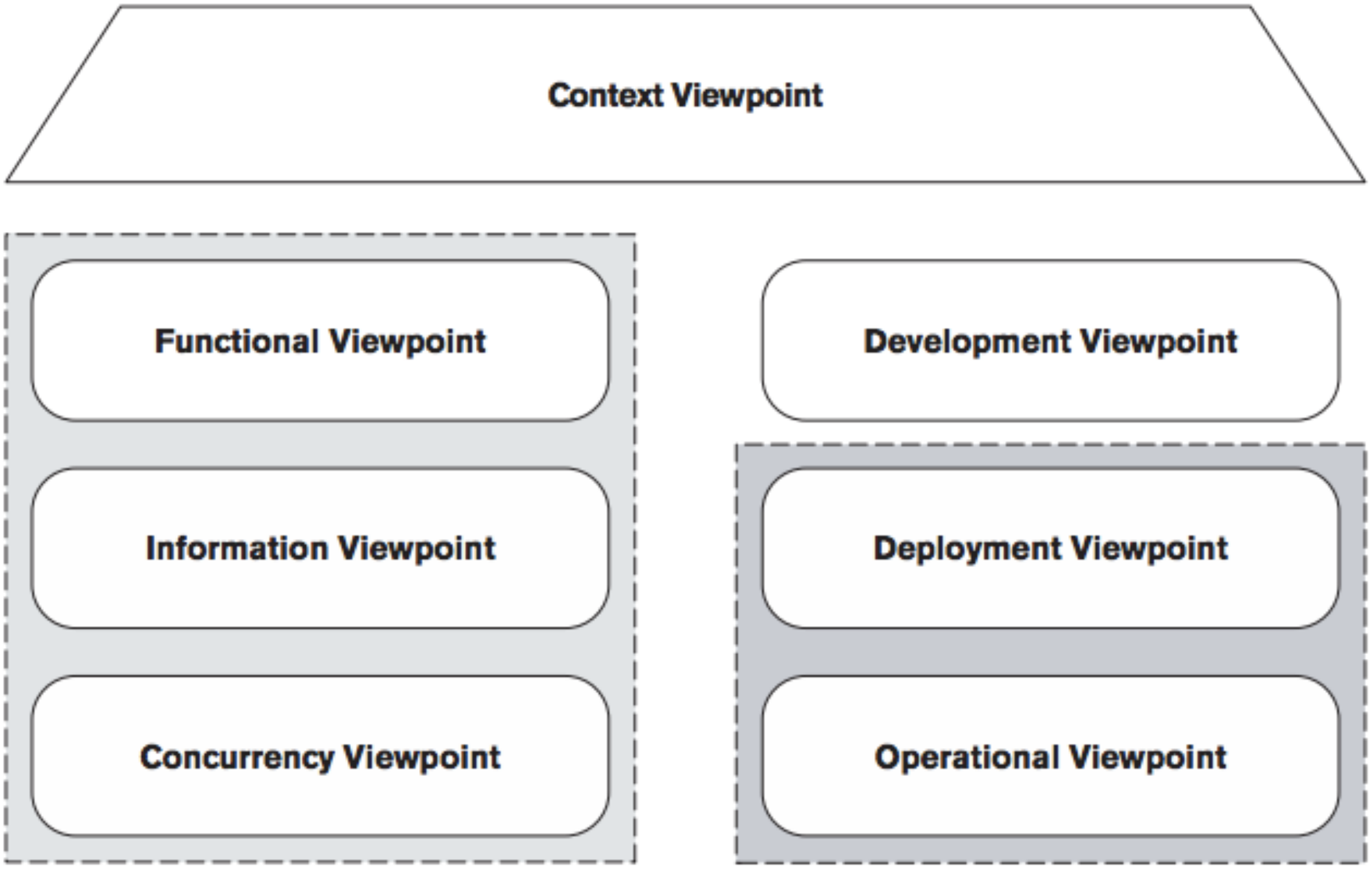
Inconsistency

Selection of the wrong set of views

Fragmentation

[SSA]

Viewpoint **Catalog**



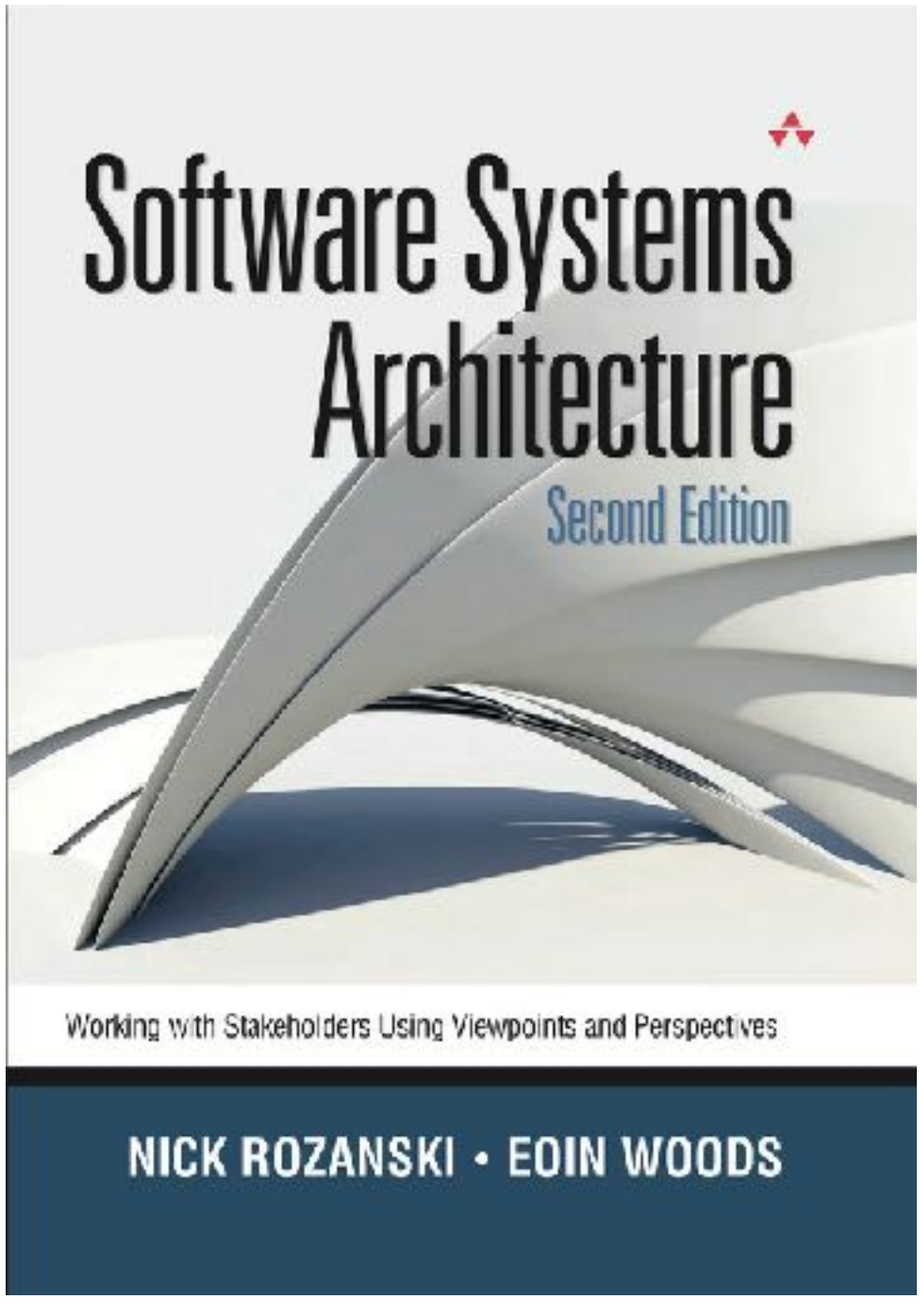
[SSA]

Viewpoint **Importance**

	Information System	Middleware	High-volume website	Entreprise package
Context	+	-	~	~
Functional	+	+	+	+
Information	~	-	~	~
Concurrency	-	+	~	+/-
Development	+	+	+	+
Deployment	+	+	+	+
Operational	+/-	-	~	+

Bibliography

- Chapters
  - **#3**: Viewpoints and Views
  - **#17**: Functional Viewpoint
  - **#20**: Development Viewpoint
  - **#21**: Deployment viewpoint



[SSA, 2011]

Functional

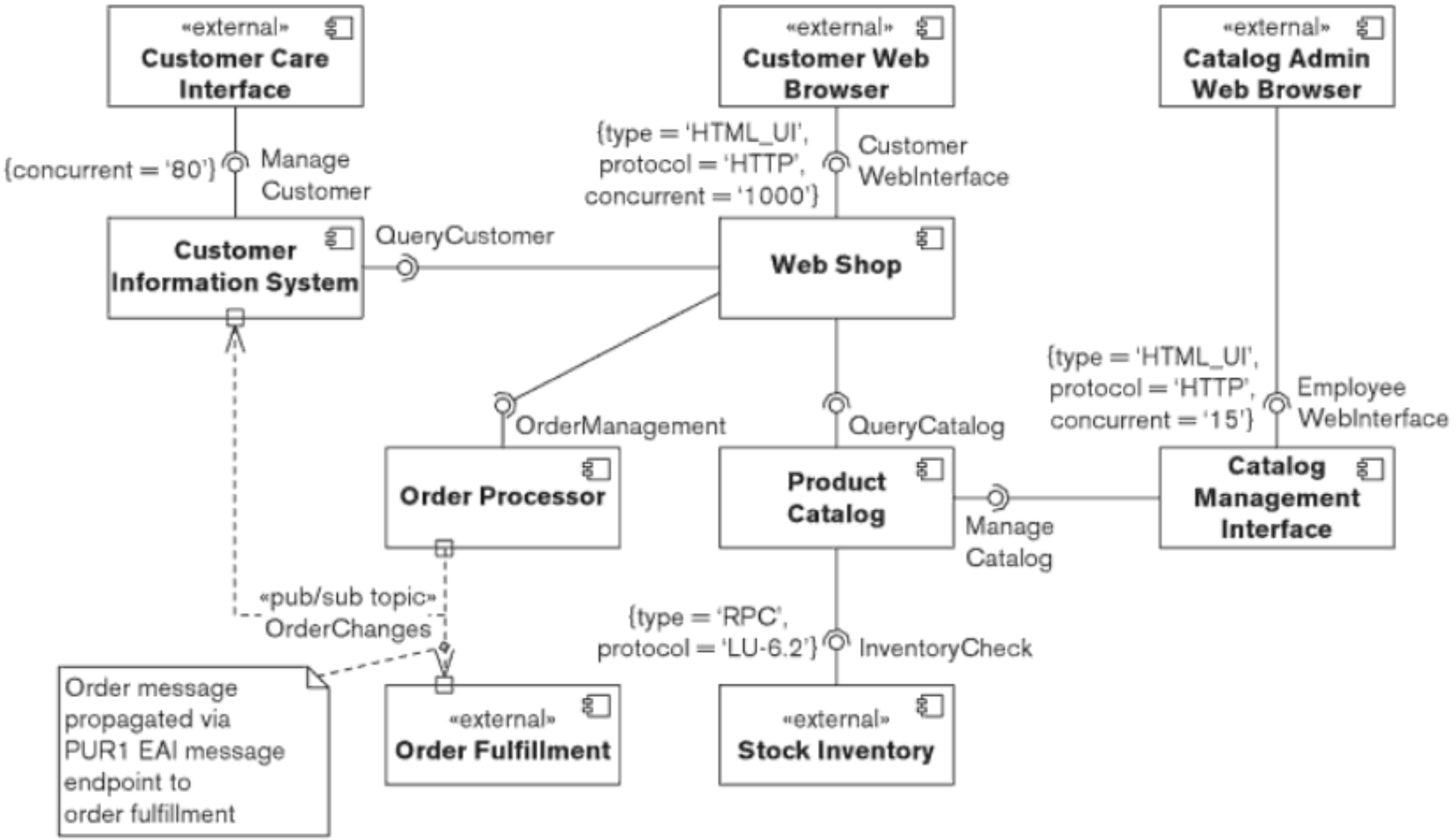
Viewpoint

Definition

> Describes the system's runtime **functional elements** and their **responsibilities**, **interfaces** and **primary interactions**

[SSA]

UML Component Diagram as a support



Elicitation process

1. Identify the elements
2. Assign responsibilities
3. Design the interfaces
4. Design the connectors
5. Check functional traceability
6. Walk through common scenarios
7. Analyse the interactions
8. Analyse for flexibility

Pitfall #1: **Poorly defined Interfaces**

Review often to assess understandability

Define interfaces and connectors **ASAP**

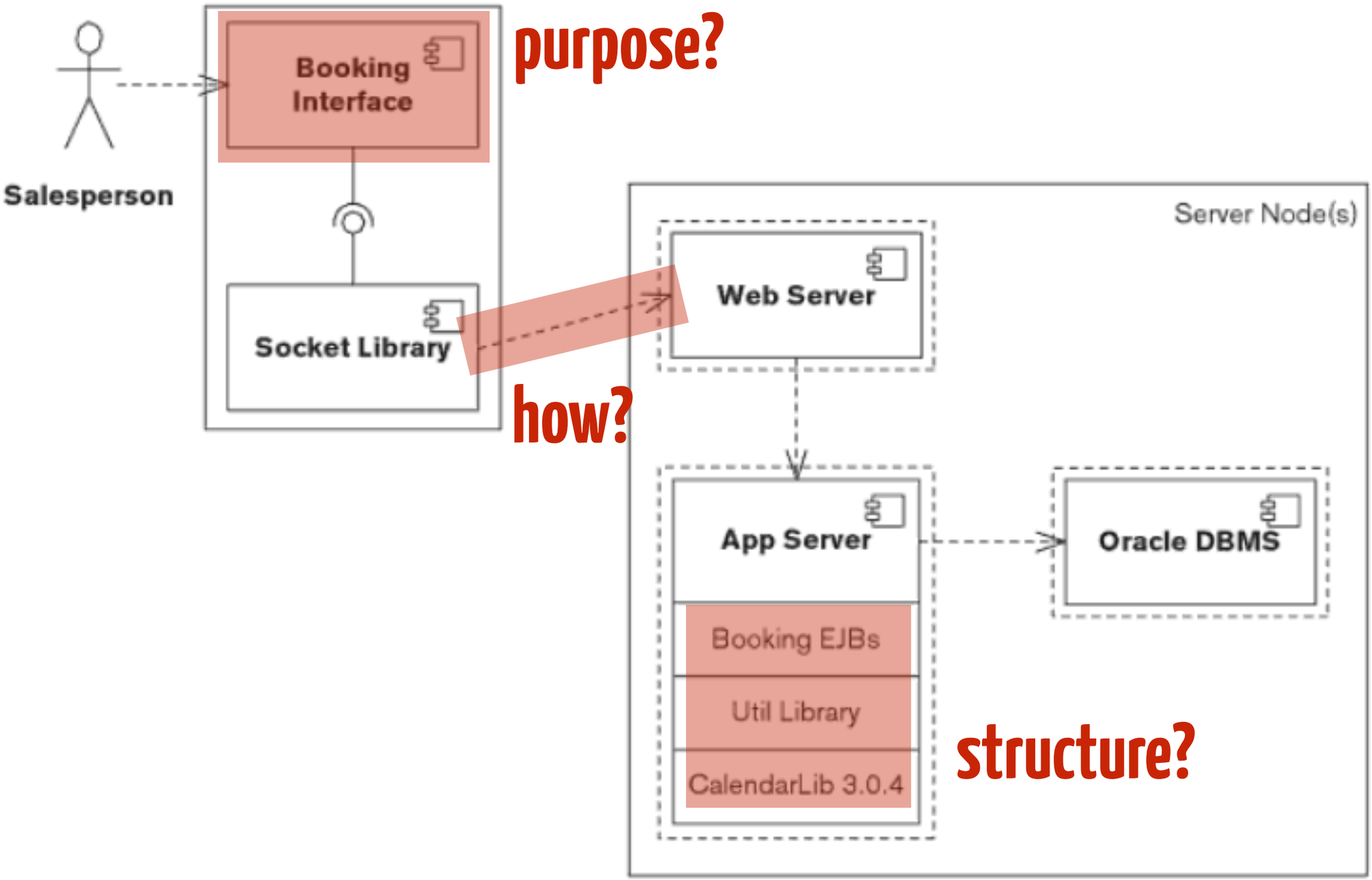
Bind **operations**, **semantics** and **examples**

Interface design ⇒ definition **completion**

[SSA]

How to avoid ?

Pitfall #2: Overloaded View



[SSA]

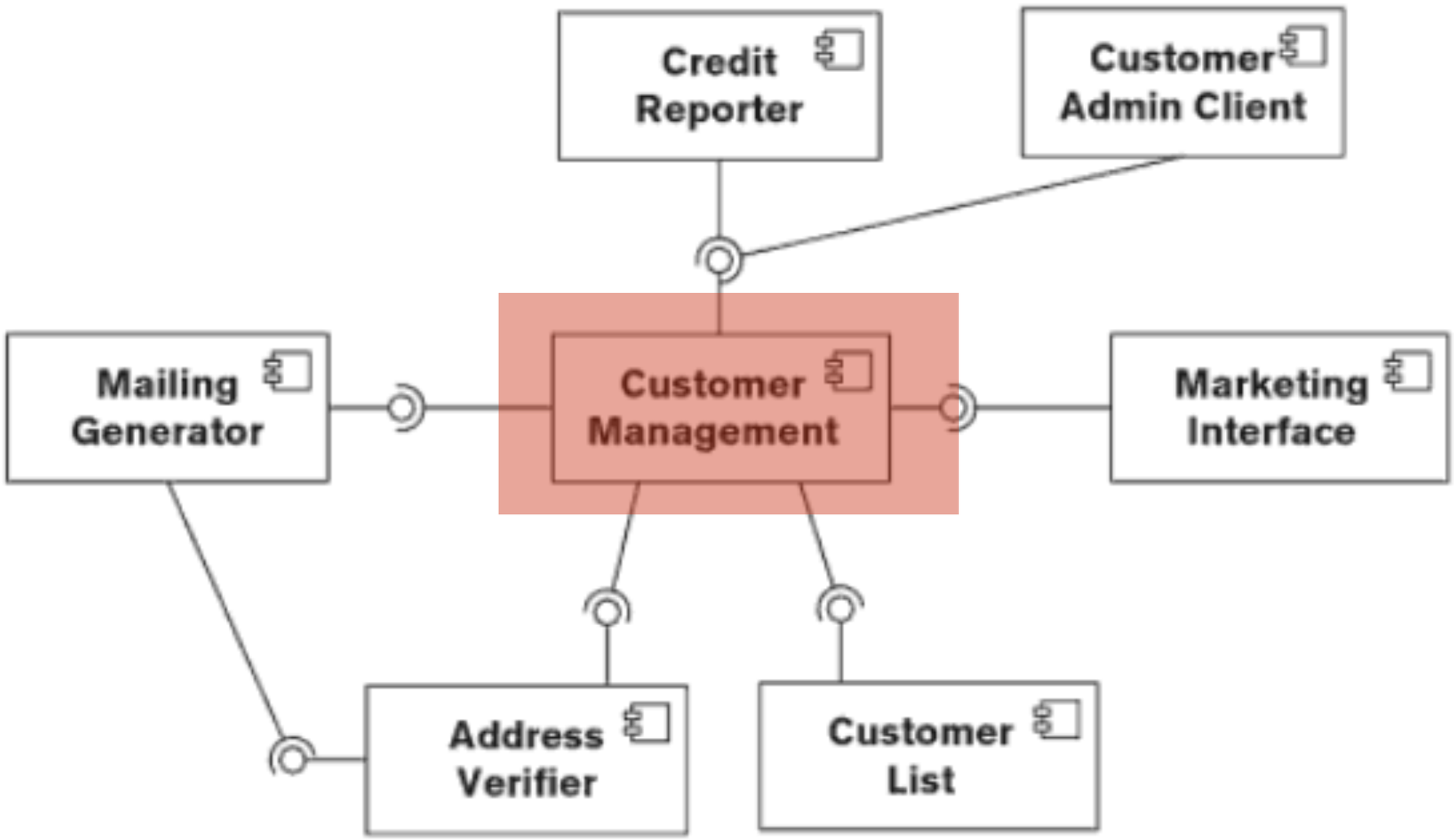
Pitfall #3: **Wrong level of details**

At most **10 elements** / view

Divide into a "**system of systems**"

[SSA]

Pitfall #4: God Element



[SSA] Contraposition: too many dependencies

How to avoid ?

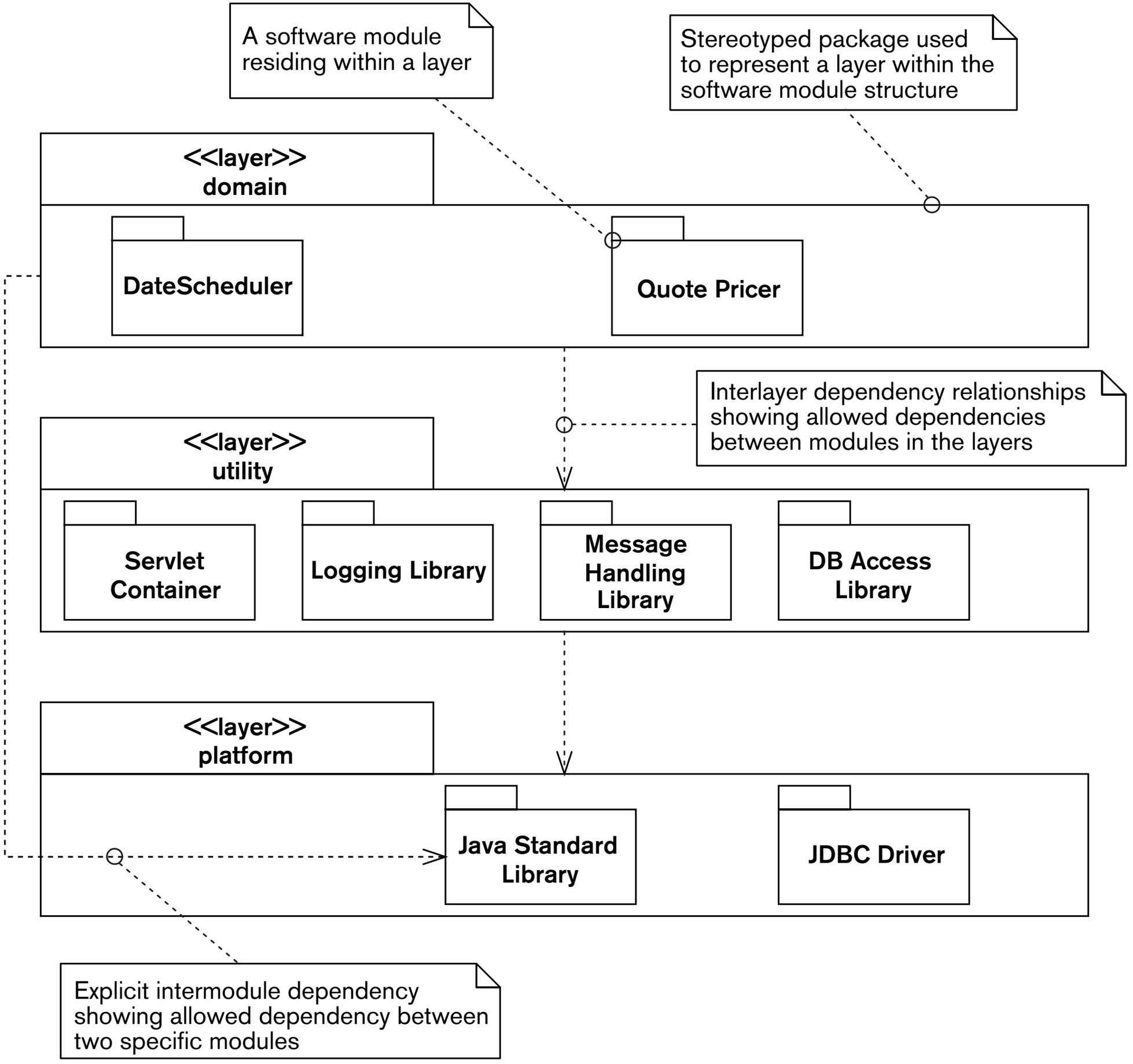
Development Viewpoint

2

Definition

> Describes the **architecture** that supports the **software development process**

[SSA]


A software module residing within a layer
Stereotyped package used to represent a layer within the software module structure
<<layer>>
domain
DateScheduler
Quote Pricer
Interlayer dependency relationships showing allowed dependencies between modules in the layers
<<layer>>
utility
Servlet Container
Logging Library
Message Handling Library
DB Access Library
<<layer>>
platform
Java Standard Library
JDBC Driver
Explicit intermodule dependency showing allowed dependency between two specific modules

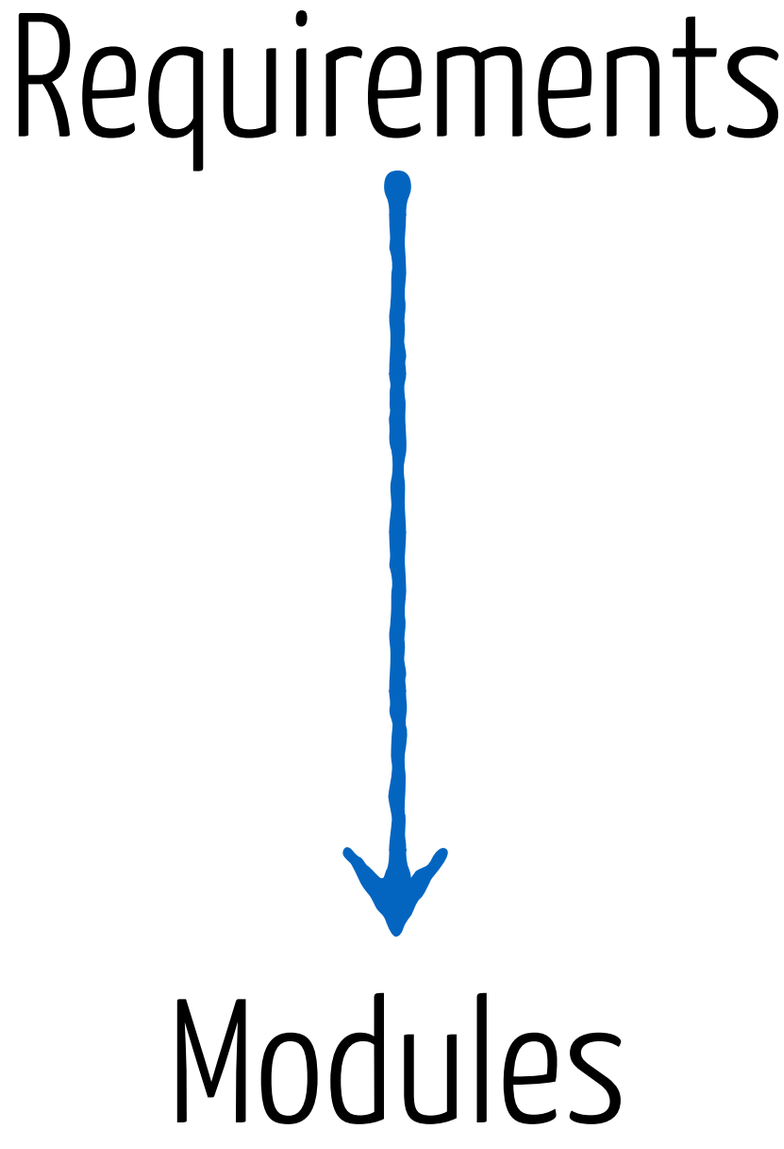
[SSA]

Elicitation process

1. Identify and classify the modules

2. Identify the dependencies

3. Identify the layering rules



Classical Pitfalls

- Too much details
- Overburdened architectural description
- Uneven focus
- Lack of developer focus
- Lack of precision
- Problem with the environment

How to avoid these pitfalls?

Deployment viewpoint

Definition

> Describes the **environment** into which the **system will be deployed** and the **dependencies** that the system has on element of it

[SSA]

IAF23 interface to production line monitoring hardware

Production Line Interface

UML nodes used to represent hardware elements within deployment environment

Primary Server
{memory = 1GB,
model = V880,
CPU = 2 x 750MHz,
mftr = Sun}

Data Capture Service

Data Access Service

Disk Array
{mftr = Sun,
model = StorEdge4500}

SCSI connection, not network

Database Server
{model = E420R,
memory = 2GB,
mftr = Sun,
CPU = 2 x 450MHz}

Oracle DBMS Instance

Internode relationships show required interconnection paths

Calculation Server
{model = V880,
mftr = Sun,
memory = 1GB,
CPU = 4 x 1.1GHz}

Predictive Calculator

Production Line Operator PC
{CPU = 750MHz,
memory = 128MB}

Operator Client

Production Planner PC
{memory = 512MB,
CPU = 1.2GHz}

Planner Client

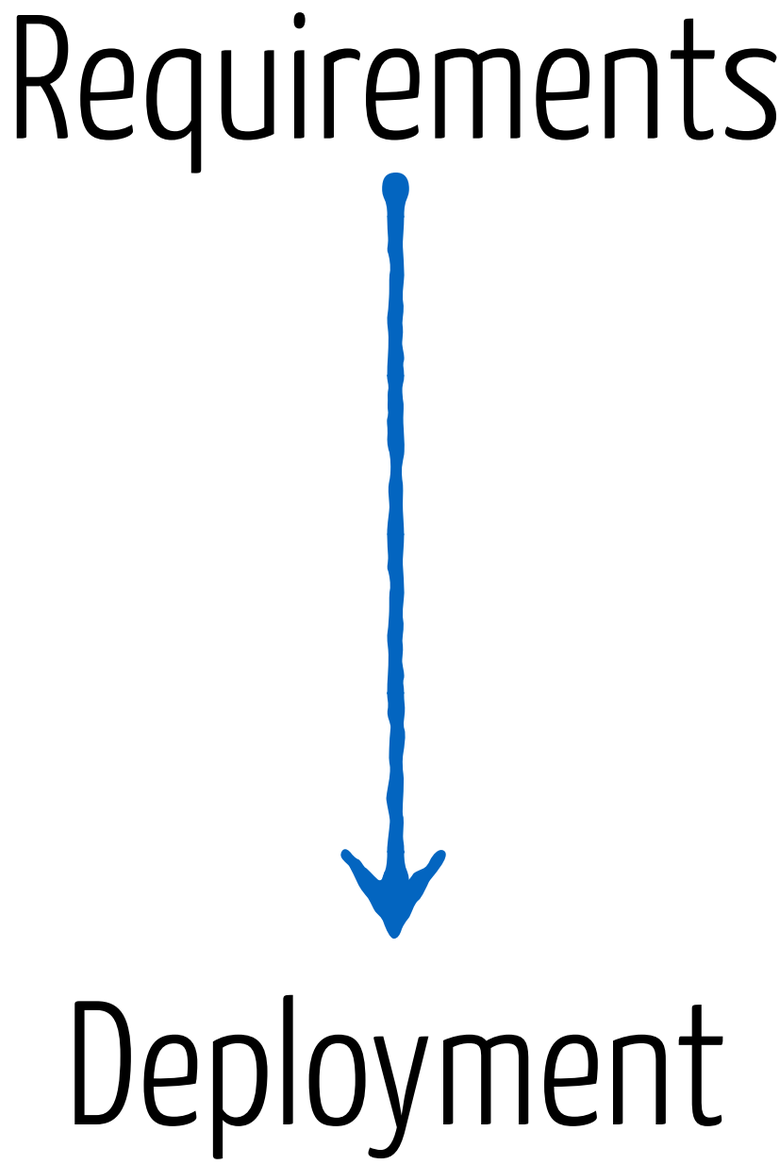
Functional elements mapped to hardware nodes

Attributes used to indicate required hardware specifications

[SSA]

Elicitation process

1. Design the deployment environment
2. Map the element to the hardware
3. Estimate the hardware requirements
4. Conduct a technical evaluation
5. Assess the constraints



Classical Pitfalls

- Unclear / Inaccurate dependencies
- Unproven technology
- Unsuitable Service-level agreement
- Lack of technical knowledge
- Late consideration of the environment
- Not specifying a disaster recovery environment

How to avoid these pitfalls?