Design and Architecture

HUSZERL Gábor
huszerl@mit.bme.hu

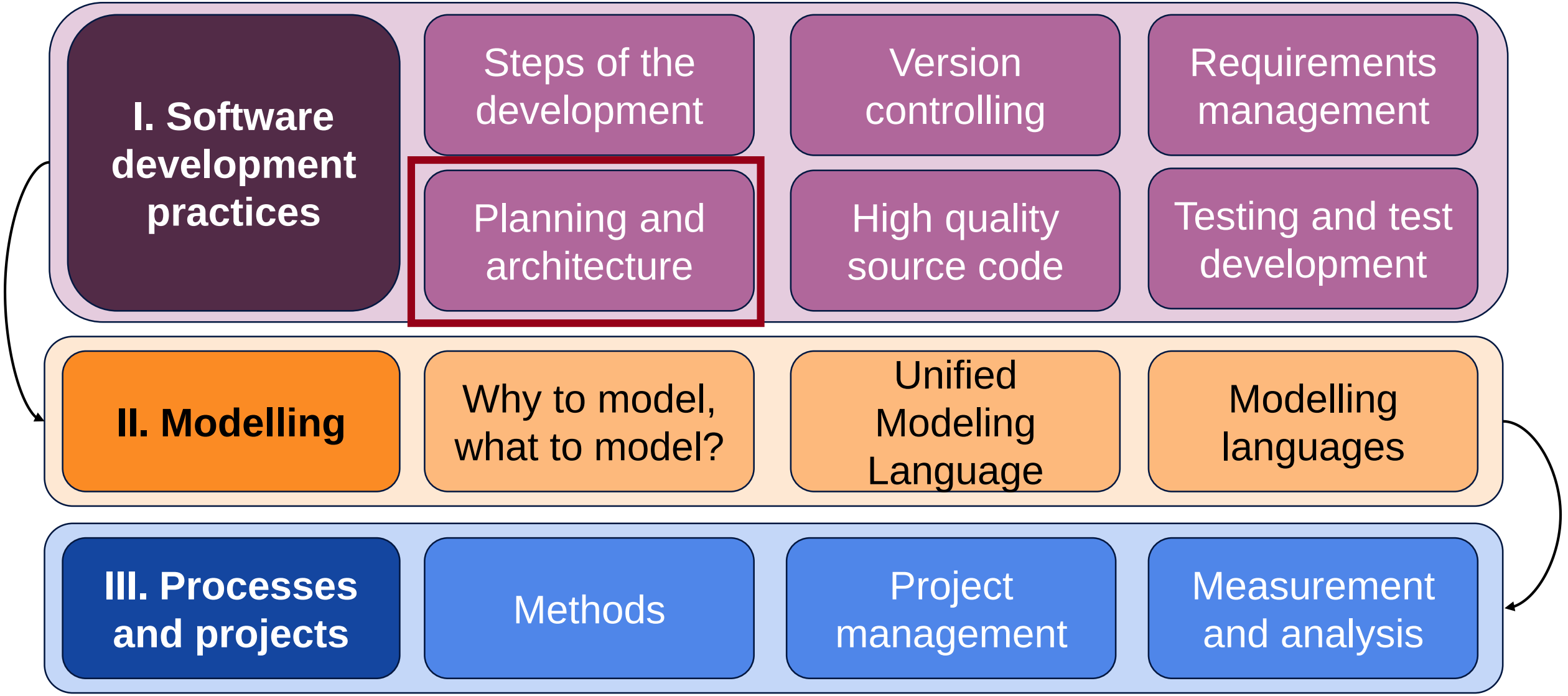
MŰEGYETEM 1782
BME VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
Méréstechnika és Információs Rendszerek Tanszék
ftsrg Critical Systems Research Group

Learning Outcomes

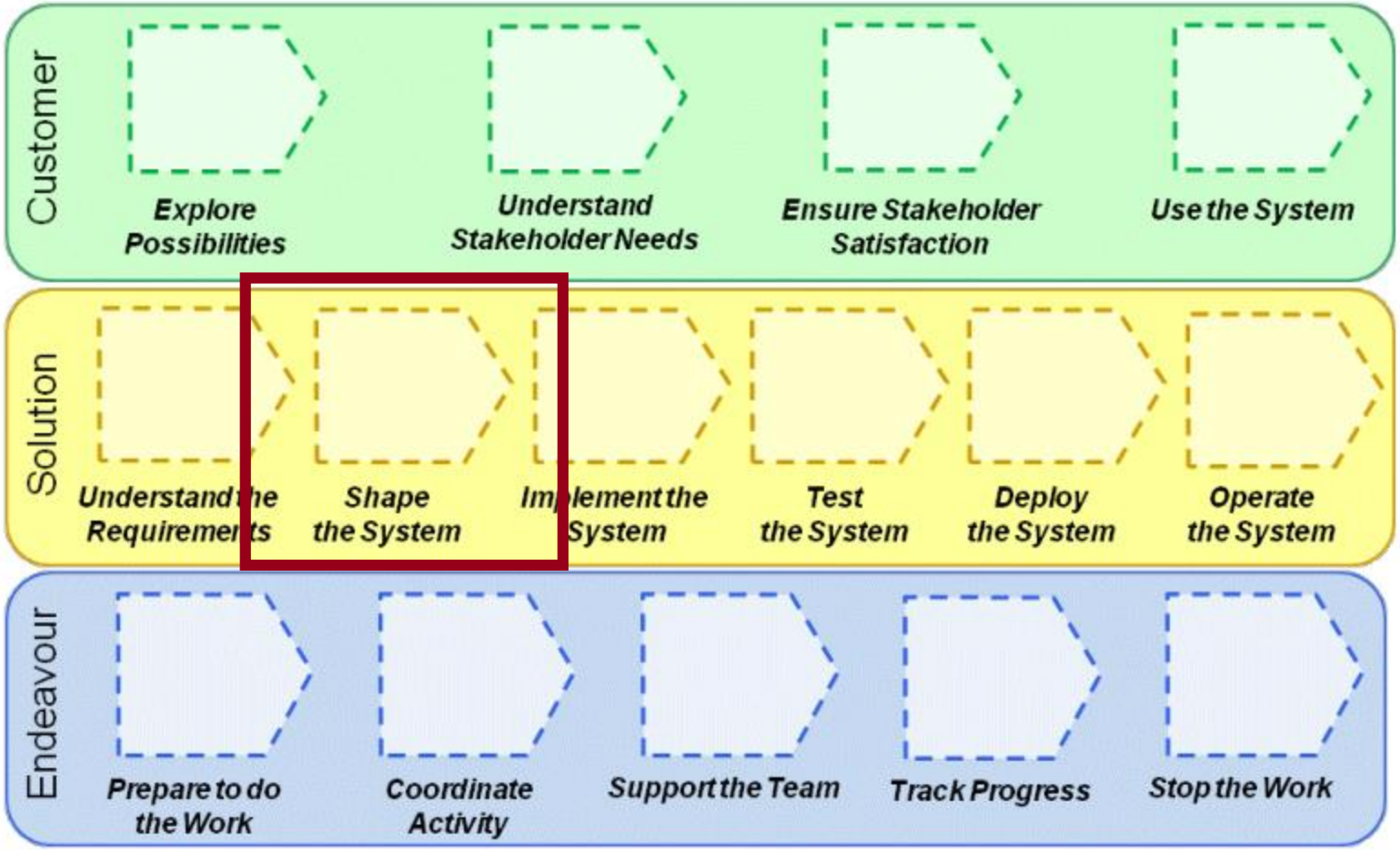
At the end of the lecture the students are expected to be able to

- (K1) define the goals of software design and software architecture,
- (K2) summarize the basic principles of software design,
- (K1) list the different software architecture elements and styles.

Further Topics of the Subject



Essence: What Can Be Done on Them?



Design

> **design**. „[process] to define the architecture, system elements, interfaces, and other characteristics of a system or system element” (IEEE)

An iterative, multi-step process

Evaluation of design alternatives

Documenting design decisions

Example: Levels of the Design Process

Requirement

Requirements of the New Application Management System (NAMS)

- *RQ1* Applicants must be able to register with their email address.
- *RQ2* Based on the selected application type, the NAMS should generate the application form filled in with data.
- *RQ3* The applicant must be able to upload the document, validly signed by the Identification-based Document Authentication (IBDA) service, to the NAMS.
- …

Example: Levels of the Design Process

Requirement

System Design

Requirements of the New Application Management System (NAMS)

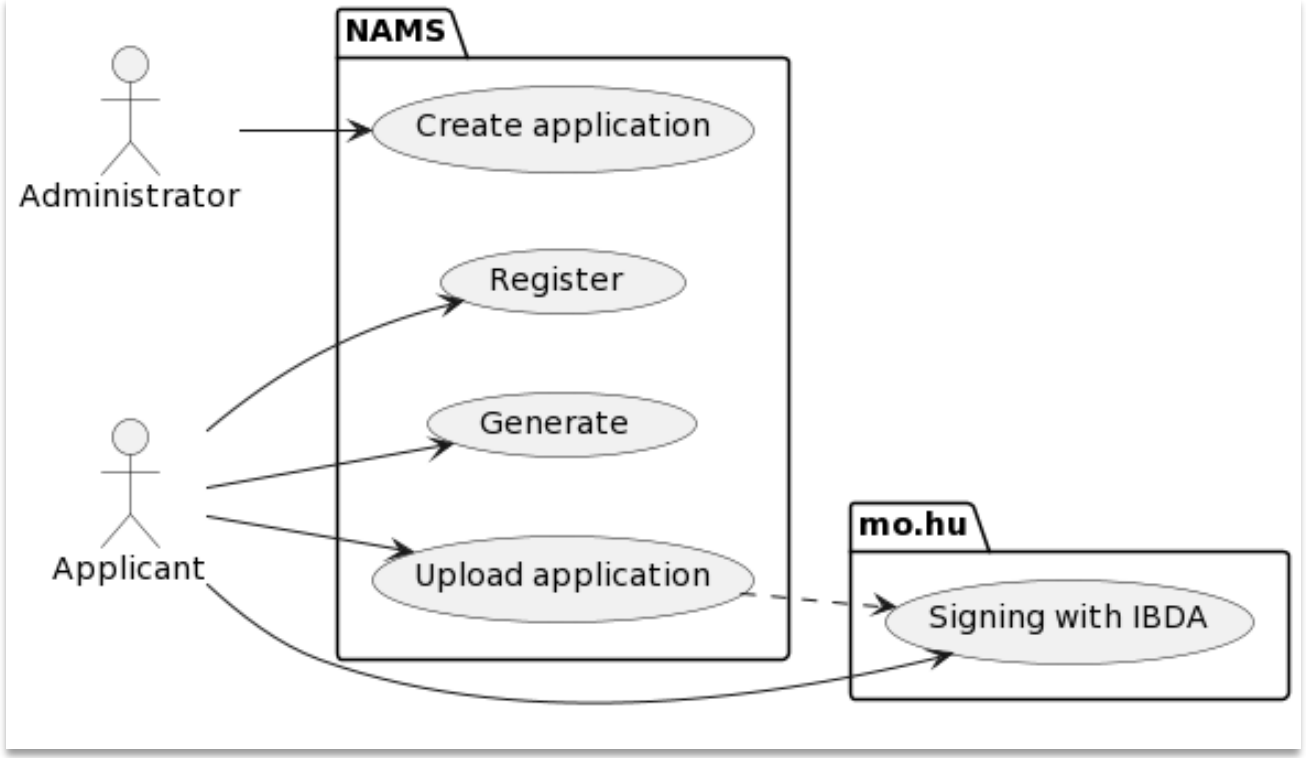
ess.

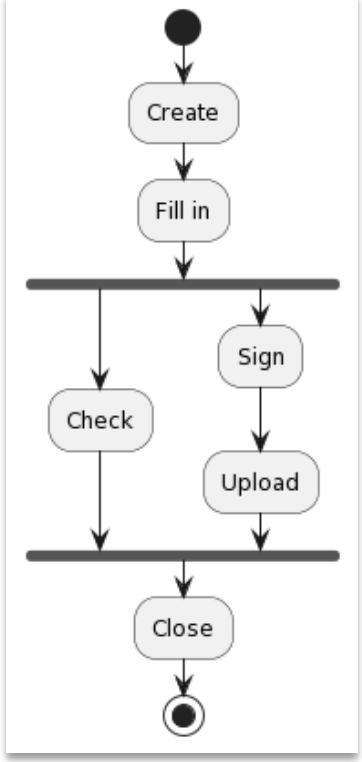
ld

idly

IBDA)

NAMS use cases and processes





Example: Levels of the Design Process

Requirement

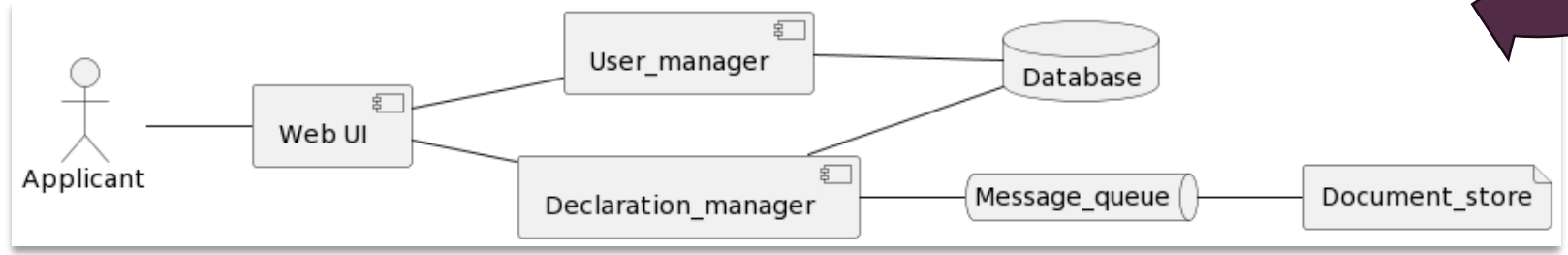
System Design

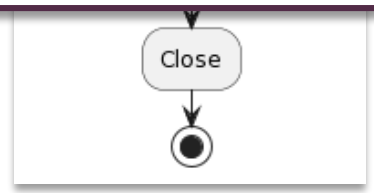
Architecture

Requirements of the New Application Management System (NAMS)

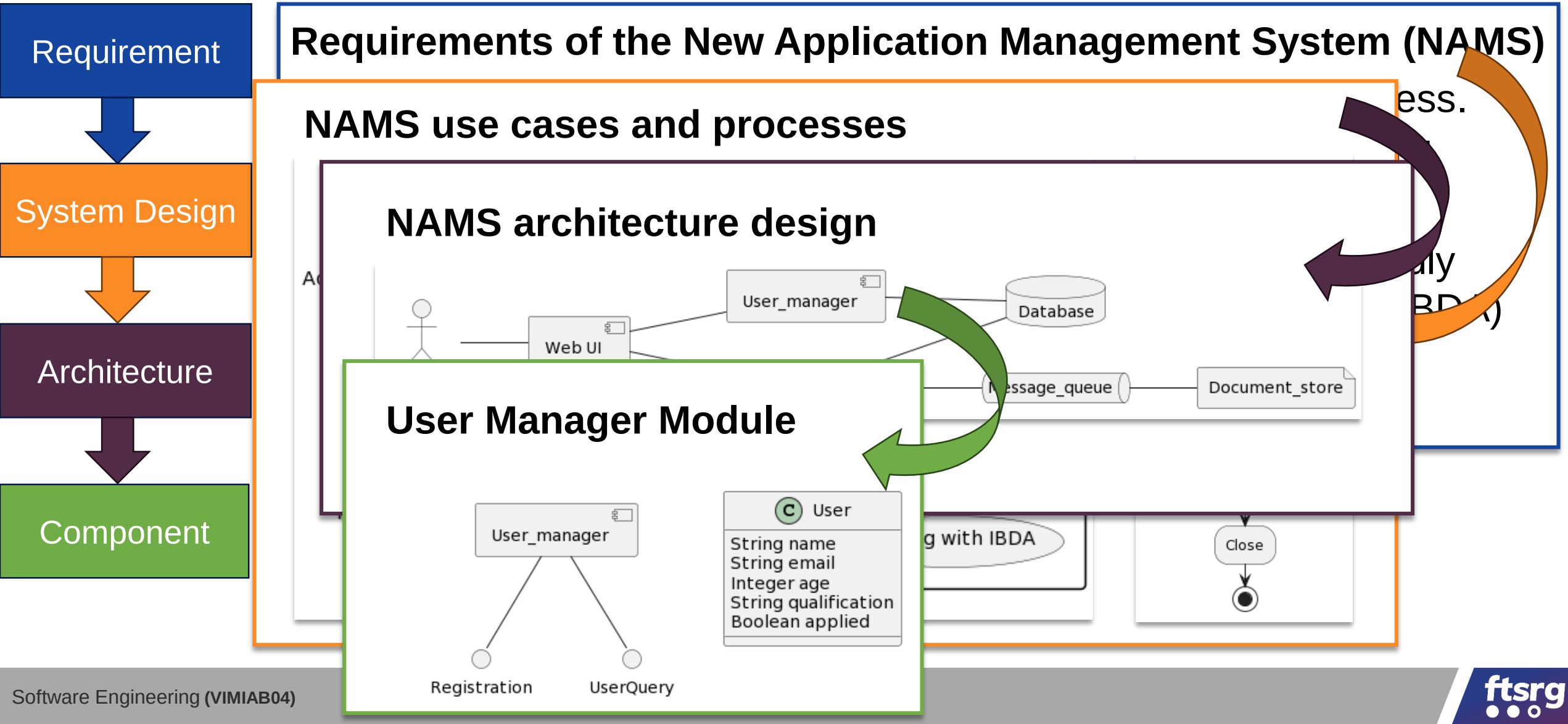
NAMS use cases and processes

NAMS architecture design





Example: Levels of the Design Process



Architecture

> **architecture.** “fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” (IEEE)

> “the decisions you wish you could get right early in a project”
>
> (Ralph Johnson / Martin Fowler)

Difficult to change later

Defines many skills (-ilities)

Break down and refine the system to meet requirements

Design Principles

Abstraction, modularization

What Features Do We Want?

A good architecture supports:

Managing complexity, clarity

Easy adaptability, extensibility

General design principles and methods

Abstraction

Decomposition and modularization

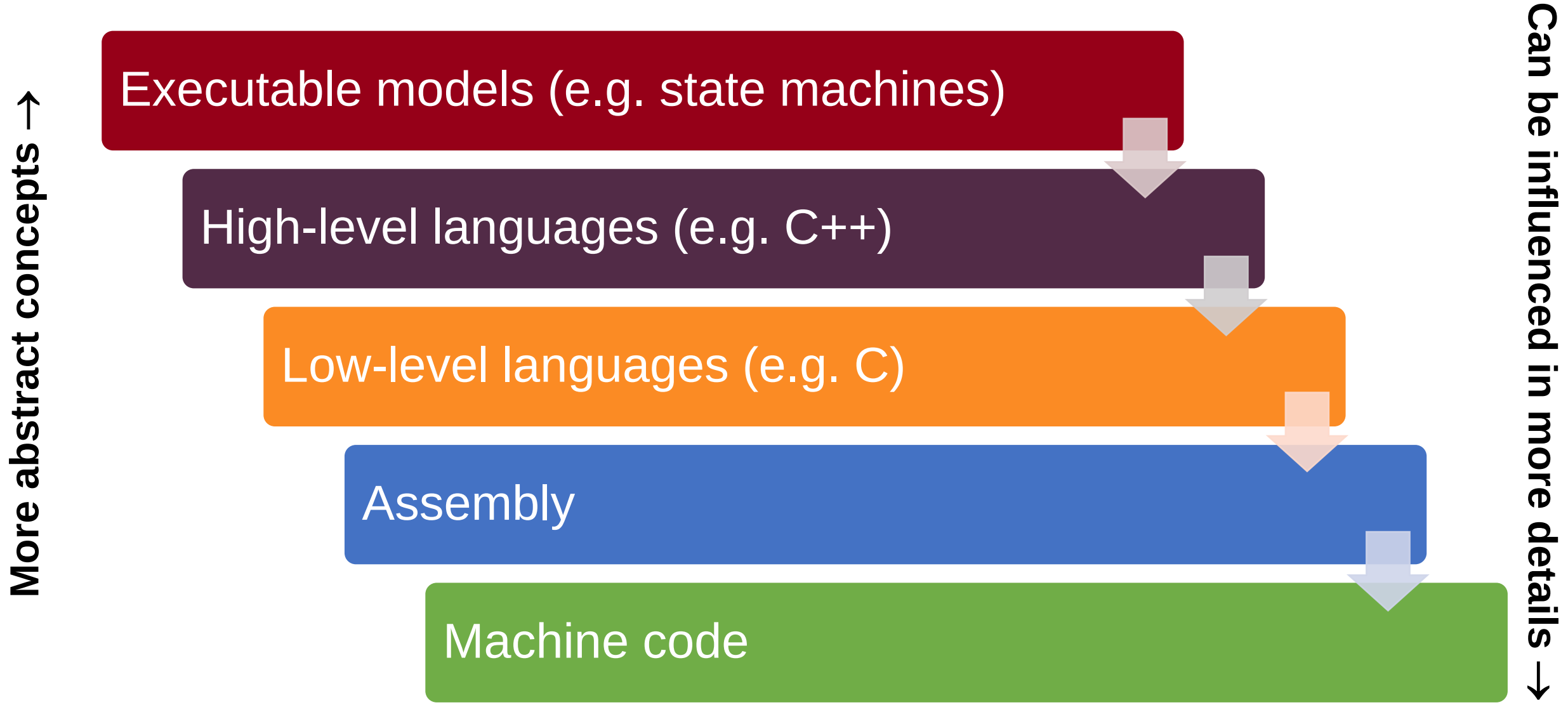
Loose coupling

Abstraction

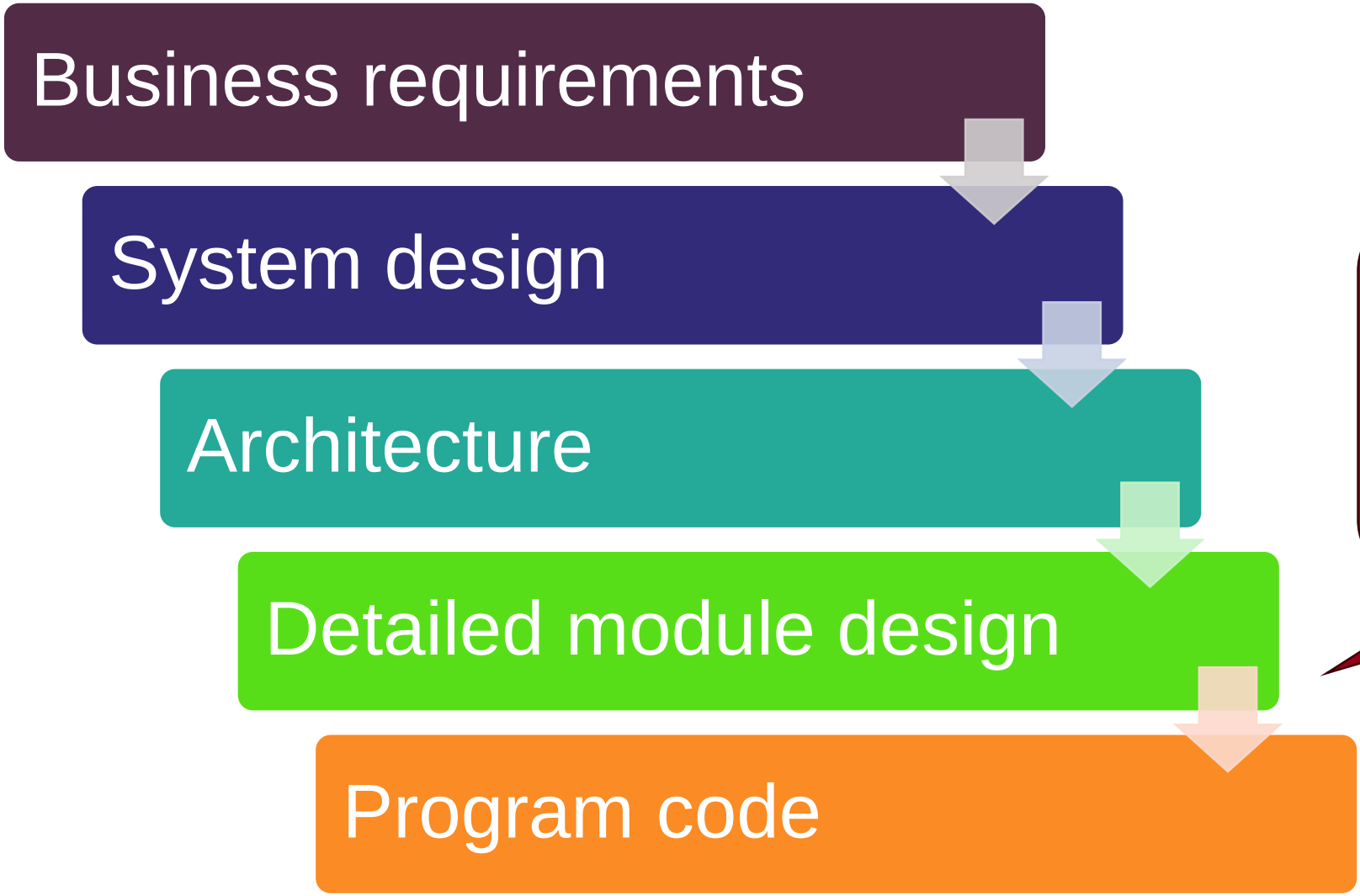
> **abstraction.** “view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information” (IEEE)

- Basic principle of modelling
  - focusing on relevant, neglecting irrelevant
- Thinking in different levels of abstraction
- Key: finding the right level for the problem

Abstraction Levels in Our Tools



Abstraction During the Development



It is not good if they are at the same level of abstraction, because then there is no design, only translation

Decomposition and Modularization

> **modular decomposition.** „process of breaking a system into components to facilitate design and development;” (IEEE)

- Well defined interfaces
  - Application Programming Interface (API)
- Information hiding
- Applied on many levels:
  - System → components (HW, SW, even on different computers)
  - Software program → classes

Different Levels of Modularization

- Procedural functions
- OO classes, interfaces
- Library with stable API
- Network API (e.g. REST)

BufferedReader

```
public BufferedReader(Reader in,
                      int sz)
```

Creates a buffering character-input stream that uses an input buffer of the specified size.

Parameters:

`in` - A Reader

`sz` - Input-buffer size

Throws:

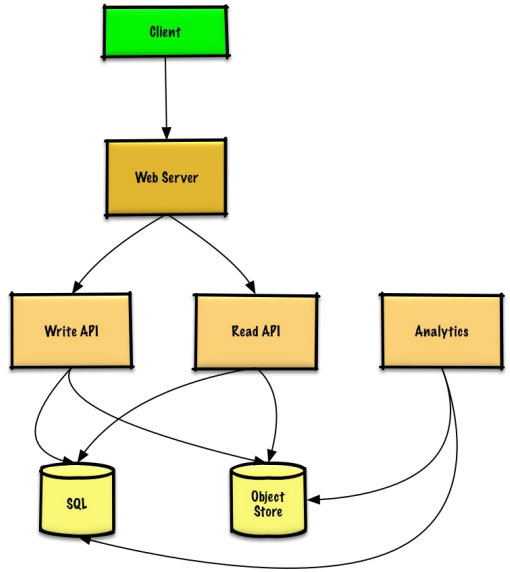
`IllegalArgumentException` - If `sz <= 0`

```
https://api.openweathermap.org/data/3.0/onecall?lat=
{lat}&lon={lon}&exclude={part}&appid={API key}
```

Parameters

`lat`	required	Latitude, decimal (-90; 90). If you need the geocoder to automatic convert city names and zip-codes to geo coordinates and the other way around, please use our Geocoding API
`lon`	required	Longitude, decimal (-180; 180). If you need the geocoder to automatic convert city names and zip-codes to geo coordinates and the other way around, please use our Geocoding API

Software Architecture



Source

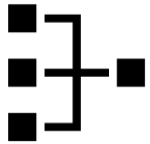
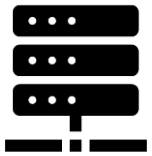
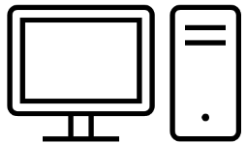
Elements, styles, challenges

Architectural Style

> **architectural style.** „a specialization of element and relation types, together with a set of constraints on how they can be used” [SWEBOK]

- General structures (layers, pipes, filters, …)
- Distributed systems (client-server, 3-layer, micro services, …)
- Interactive systems (MVC: Model-View-Controller, …)

Example: Basic Client-Server Elements

Client	Network	Server		
		Presentation	Business logic	Data
Mobile application 	**Internet** 			**Database** 
Web based application (frontend) 	**Enterprise network** 	**Web server** 	**Application server** 	**File/object store** 
Thick client application (desktop) 				

See courses „Mobile and Web Based Software”, „Communication Networks”, „Software Techniques”

Why are such systems difficult to design?

Distributed systems

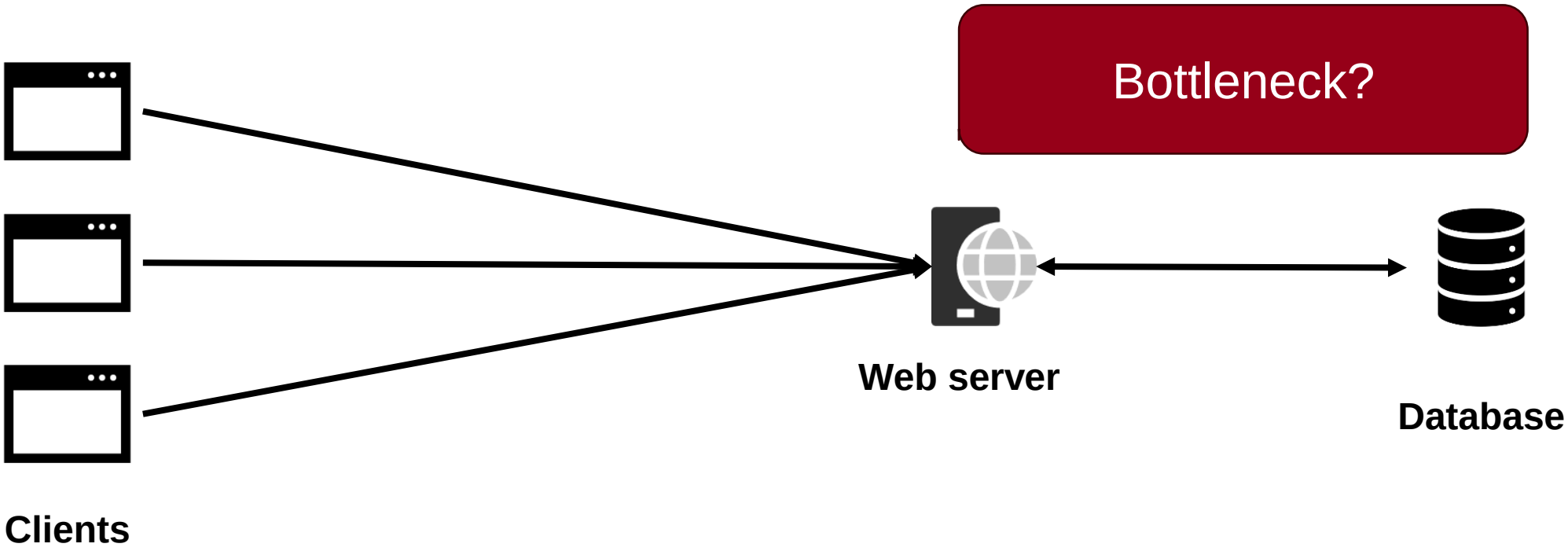
> „A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable” (Leslie Lamport)

- Concurrent operation
- Lack of a global picture
- High complexity

See partially in course „Operating systems”

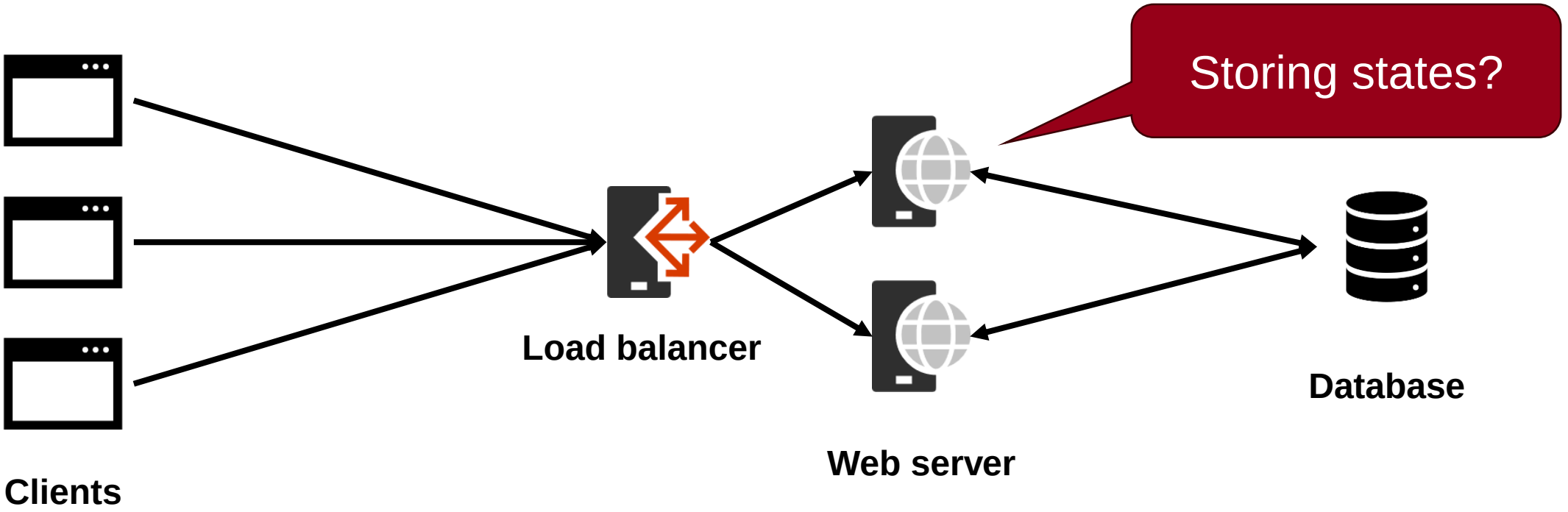
Challenge: Scalability?

~ If we put more resources into the system, will its performance increase proportionally?



Challenge: Scalability?

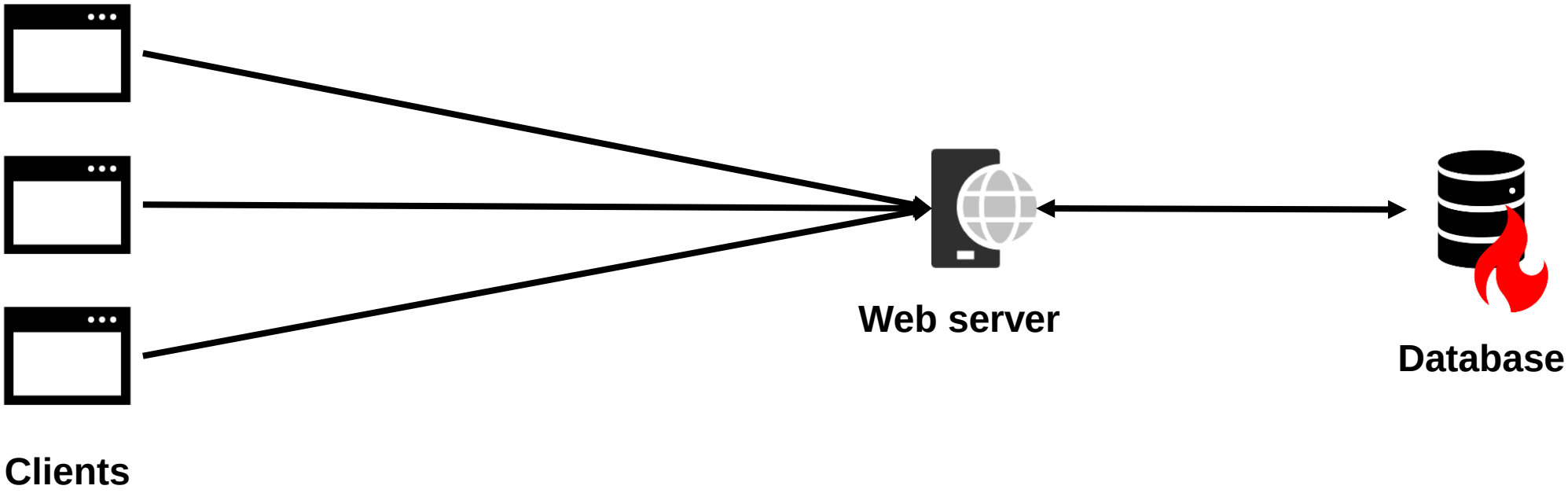
~ If we put more resources into the system, will its performance increase proportionally?



See e.g. MSc course „Scalable software”

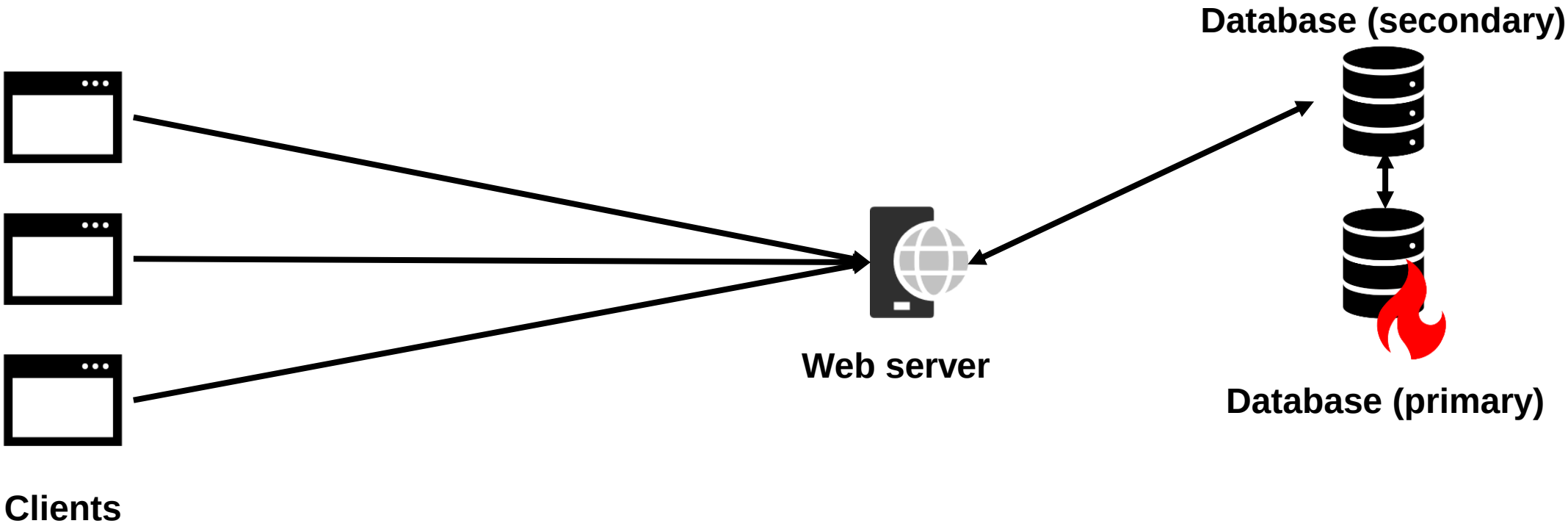
Challenge: Dependability?

~ Is the system able to provide service
in case of various HW/SW/... failures?



Challenge: Dependability?

~ Is the system able to provide service
in case of various HW/SW/... failures?

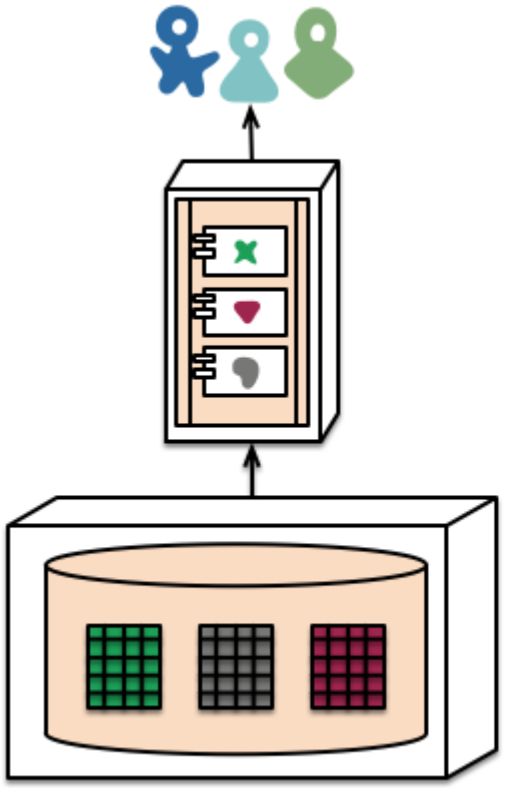


See e.g. MSc course „Dependable distributed and decentralized systems”

Example: Architecture Style

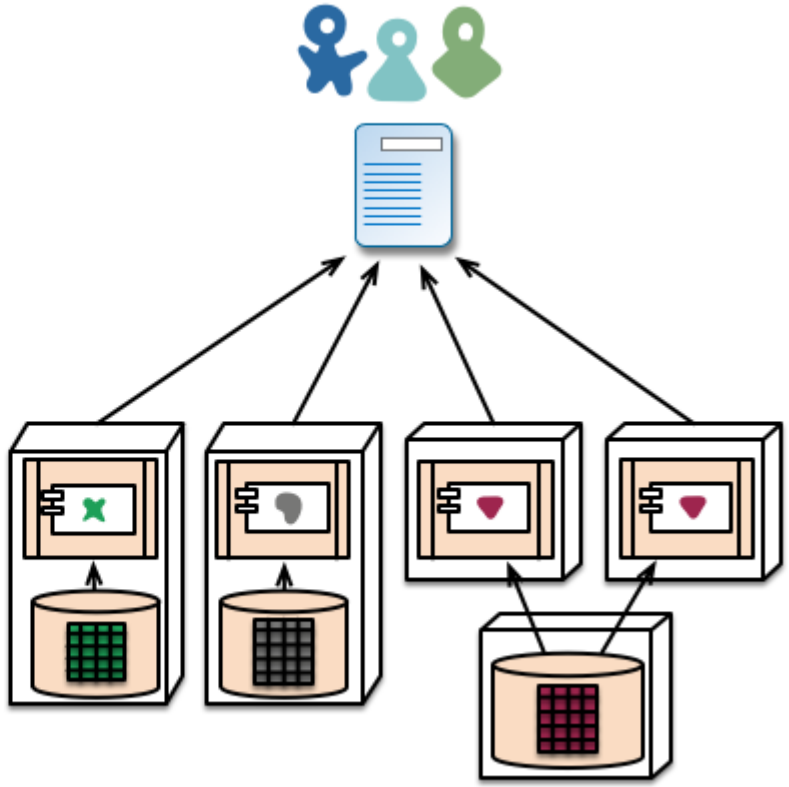
Depends on …! Both of them have their advantages and disadvantages

Monolith



monolith - single database

Microservice



microservices - application databases

Further reading:
Werner Vogels,
Monoliths are not dinosaurs.
https://www.allthingsdistributed.com/2023/05/monoliths-are-not-dinosaurs.html

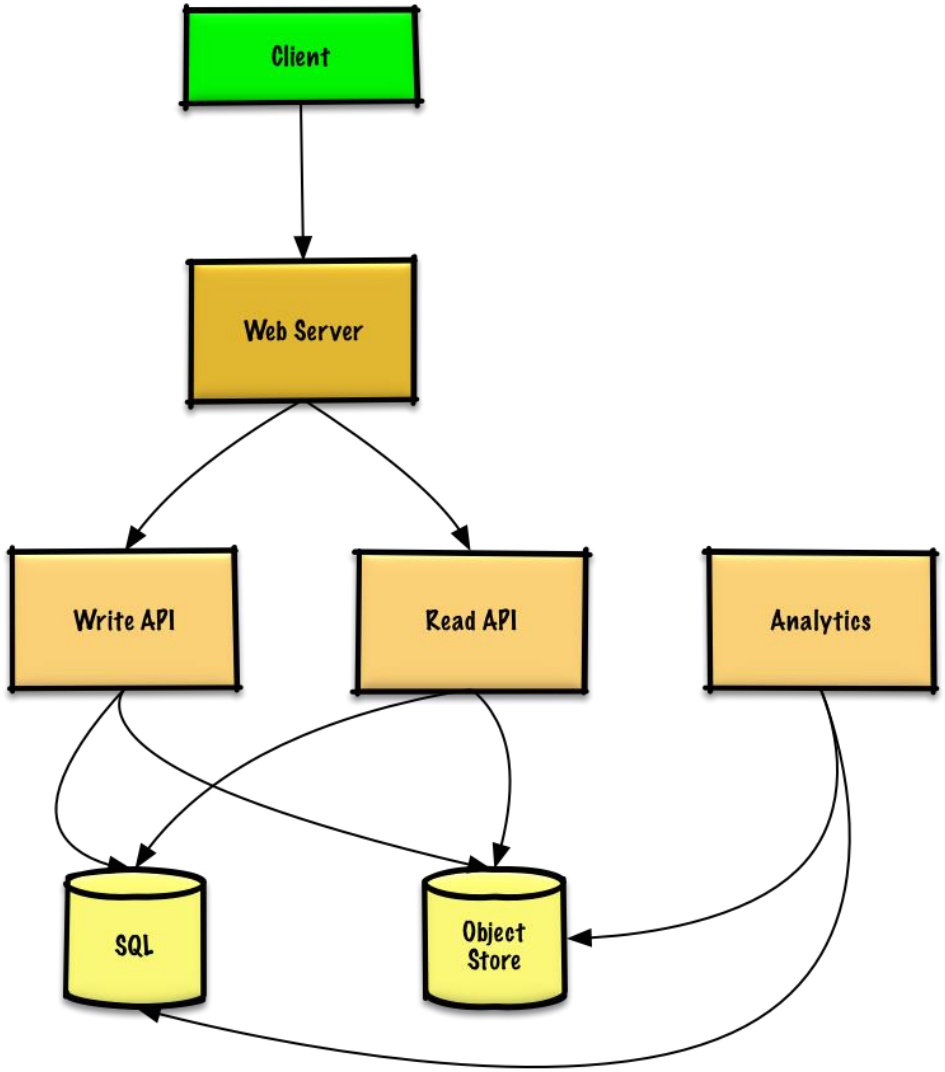
Source

Complex Architecture Design Example

- Design architecture for a system similar to pastebin.com service

- Main use case:
  - The user enters a long text and gets a generated link

- (Optional) detailed description:
  - system-design-primer

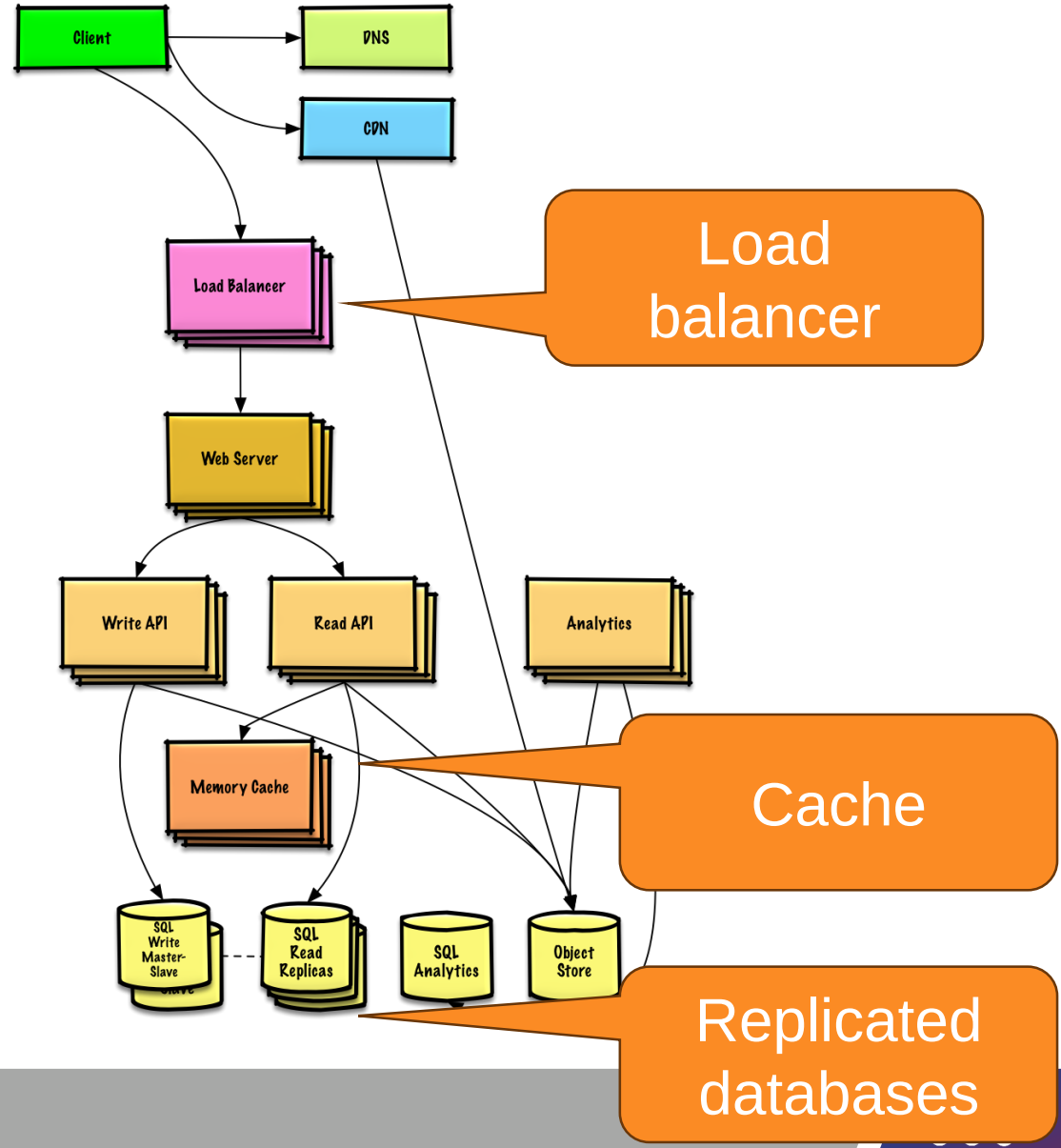


Complex Architecture Design Example

- Design architecture for a system similar to pastebin.com service

- Main use case:
  - The user enters a long text and gets a generated link

- (Optional) detailed description:
  - [system-design-primer]



Example: Documenting System Architecture

The system architecture description should do the following:

a) include traceability information to system requirements
b) establish the principles for partitioning the system into system elements (such as hardware, software and operations)
c) record the important properties of, and relationships among, those elements in a manner consistent with the work breakdown structure
d) demonstrate that architecturally-significant requirements are met and allocated to provide a framework for requirements specification and design refinement.

Traceability

Why we have decomposed it in this way?

Demonstrate the satisfaction of the requirements

Source: ISO/IEC/IEEE 15289 Std.

Detailed Design

Module/unit level design

Preparing Detailed Design

- Enterprise or critical systems
  - Preparing design for each important components, modules
  - Focus on: interfaces, defining data
  - Specifying internal operation: state machines, data flows, …
- Agile design and development
  - Less design in advance
  - Often collaborative process, brainstorming
  - Part of every day activities

Principles of Object-Oriented Design (SOLID)

How to design maintainable and reusable OO code?

- **S**ingle Responsibility: A class should have one, and only one, reason to change
- **O**pen Closed: entities ... should be open for extension, but closed for modification
- **L**iskov Substitution: Derived classes must be substitutable for their base classes
- **I**nterface Segregation: Make fine grained interfaces that are client specific.
- **D**ependency Inversion: Depend on abstractions, not on concretions.

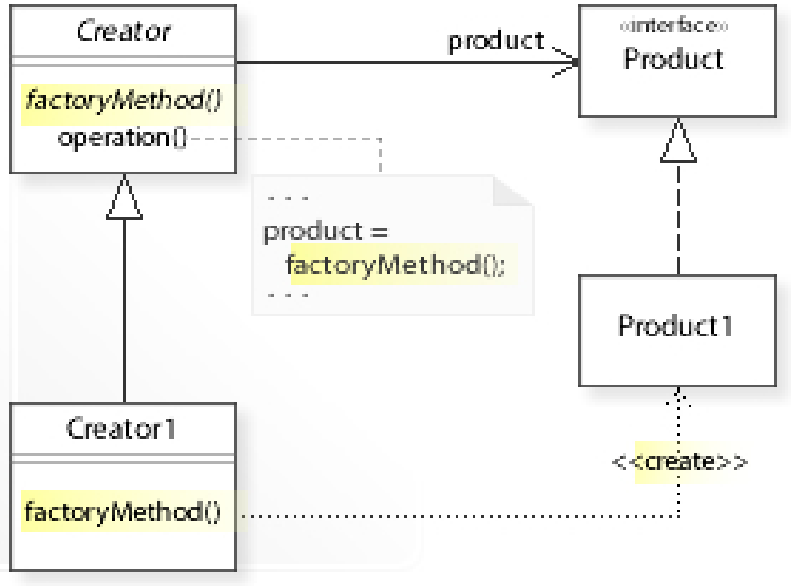
Source: Robert C. Martin

Design Patterns

design pattern. „a common solution to a common problem in a given context” (SWEBOK)

Example: Factory method

- Pattern for object creation
- How can descendants decide what to create?
- Have a separate method for this!



See in more details in course „Software Techniques”

Example: Documenting Software Design

The software design description presents the characteristics of one or more systems, subsystems, software items, or other system components, and their interfaces. It includes the following:

a) identification of external interfaces, software components, software units, and other interfaces
b) allocation of software item requirements to software components, further refined, as needed, to facilitate detail design
c) description of the items (systems, configuration items, users, hardware, software, etc.) that communicate with other items to pass and receive data, instructions or information
d) the concept of execution including data flow and control flow
e) security considerations
f) reuse elements
g) error handling.

It should include the following:

a) traceability information to both architectural components and software requirements.
b) specification of protocols
c) partitioning of the software into design entities and description of the important properties and relationships among those entities.

Defining interfaces

Traceability

Source: ISO/IEC/IEEE 15289 Std.

Summary

DISCLAIMER: These Were the Basics Only …

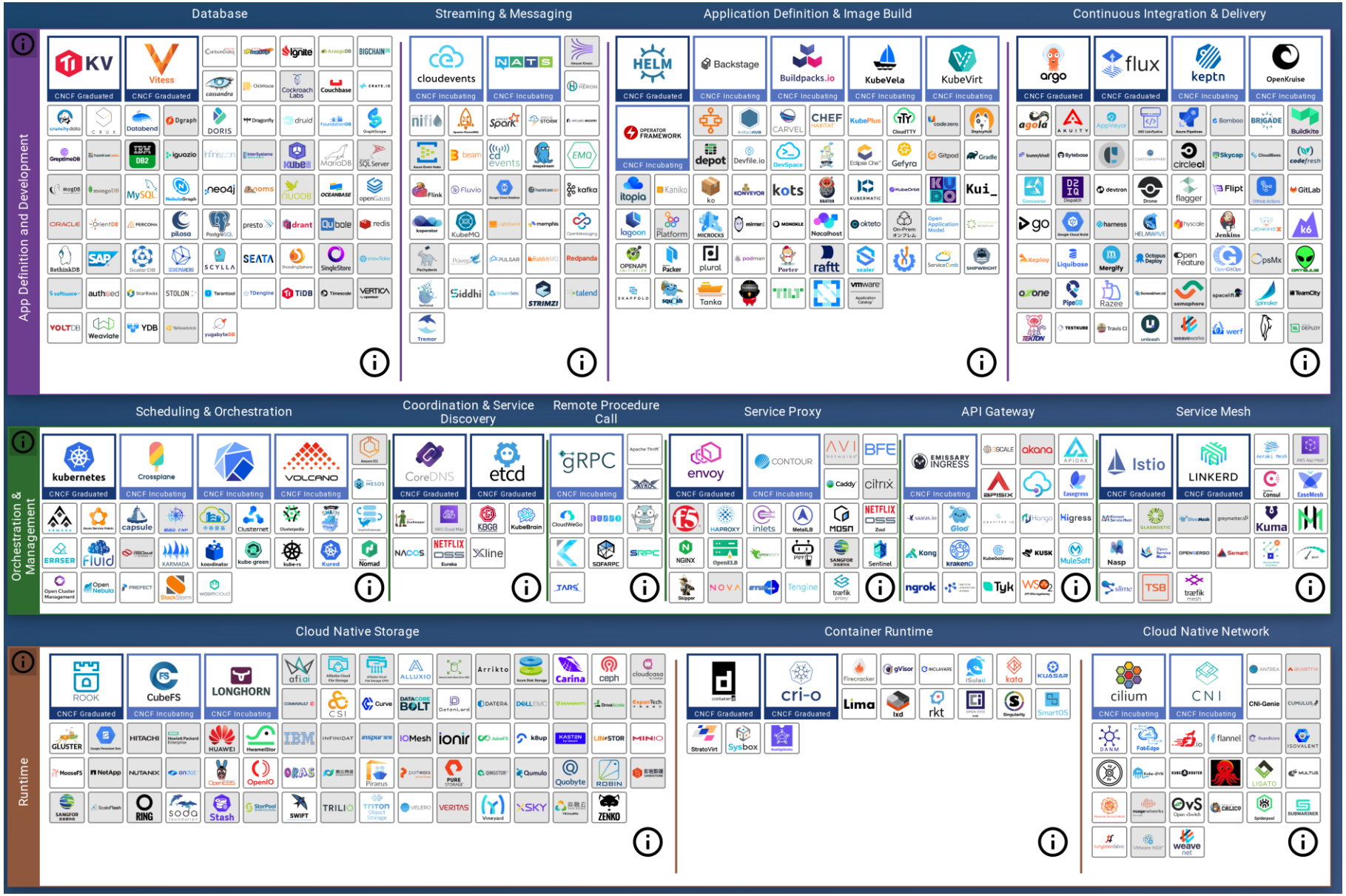
Good architecture design also requires:

- Theoretical knowledge (performance analysis, fault tolerance design, architecture patterns, consistency models, …) → **MSc**
- Knowing the platforms (relational/NoSQL/graph databases, cloud infrastructure, message queues, containerization, …) → **MSc**
- 5-10 years of experience with good and bad systems ☺

Outlook

Cloud Native Landscape

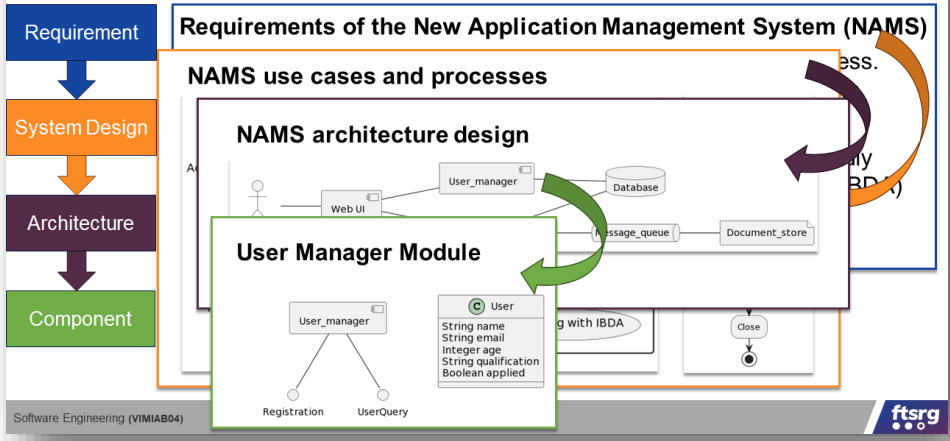
(excerpt)



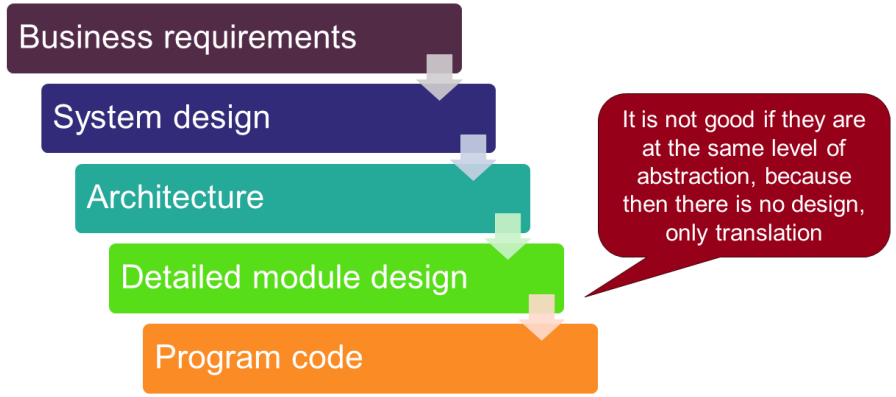
https://landscape.cncf.io/

Summary

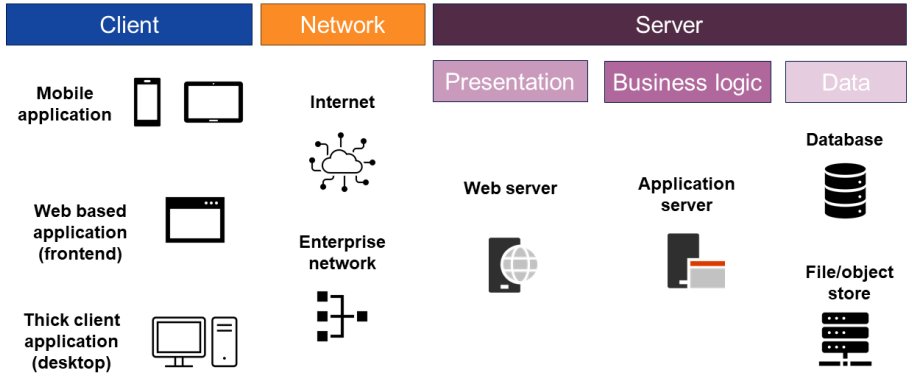
Example: Levels of the Design Process



Abstraction During the Development



Example: Basic Client-Server Elements



See courses „Mobile and Web Based Software", „Communication Networks", „Software Techniques"

Complex Architecture Design Example

- Design architecture for a system similar to pastebin.com service
- Main use case:
  - The user enters a long text and gets a generated link
- (Optional) detailed description:
  - system-design-primer