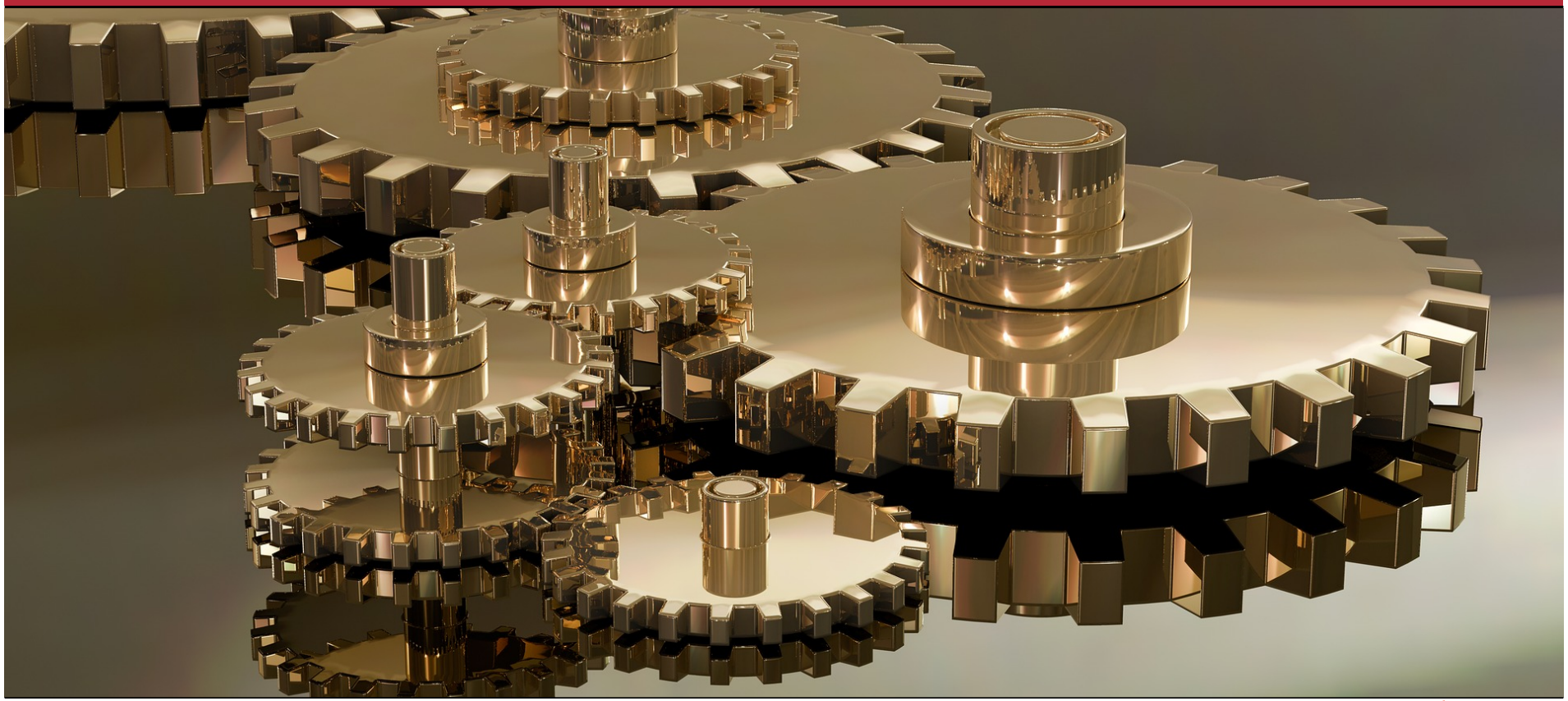
Software Engineering

TECHNISCHE UNIVERSITÄT DARMSTADT

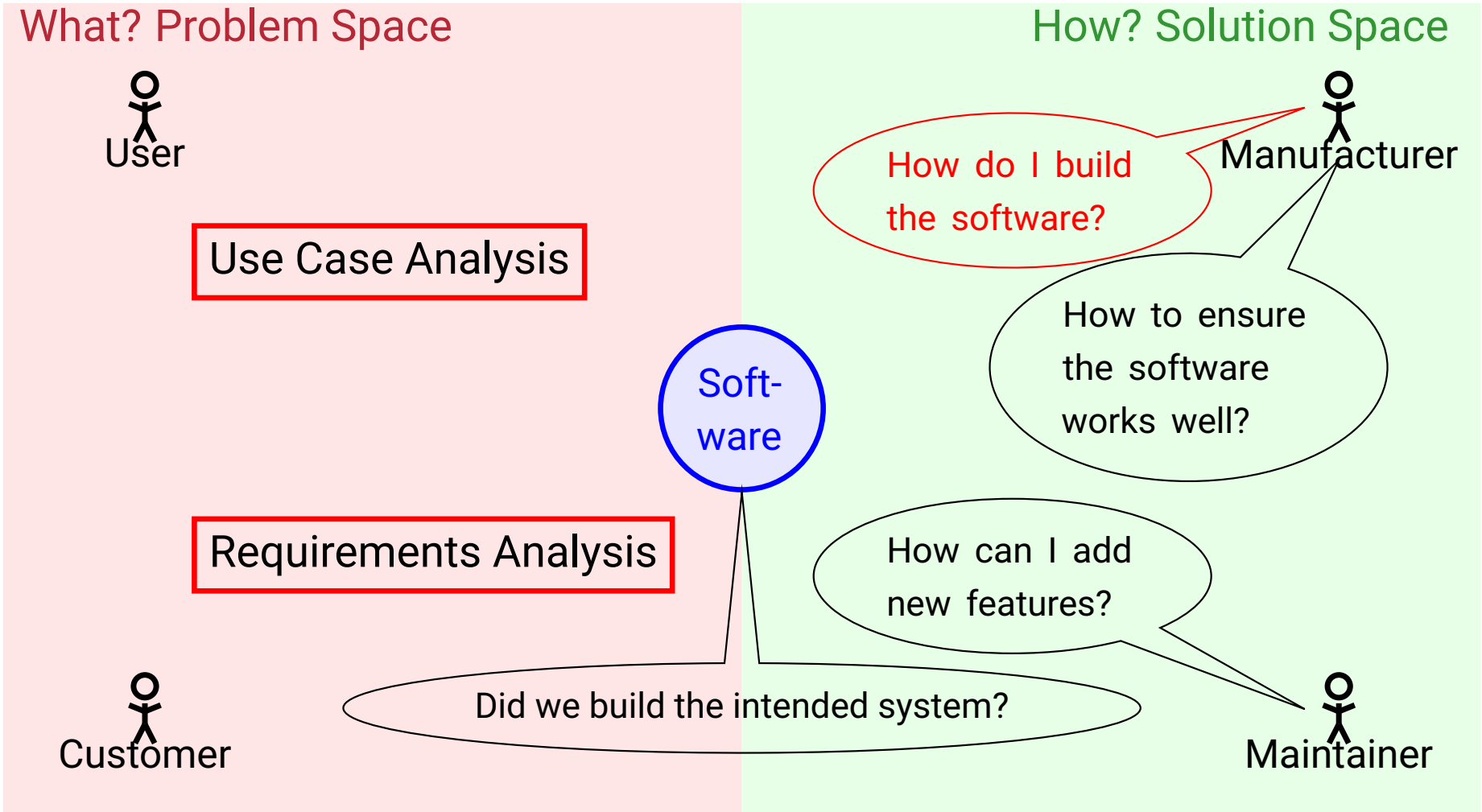
Software Architecture

Prof. Dr. Reiner Hähnle
Fachgebiet Software Engineering



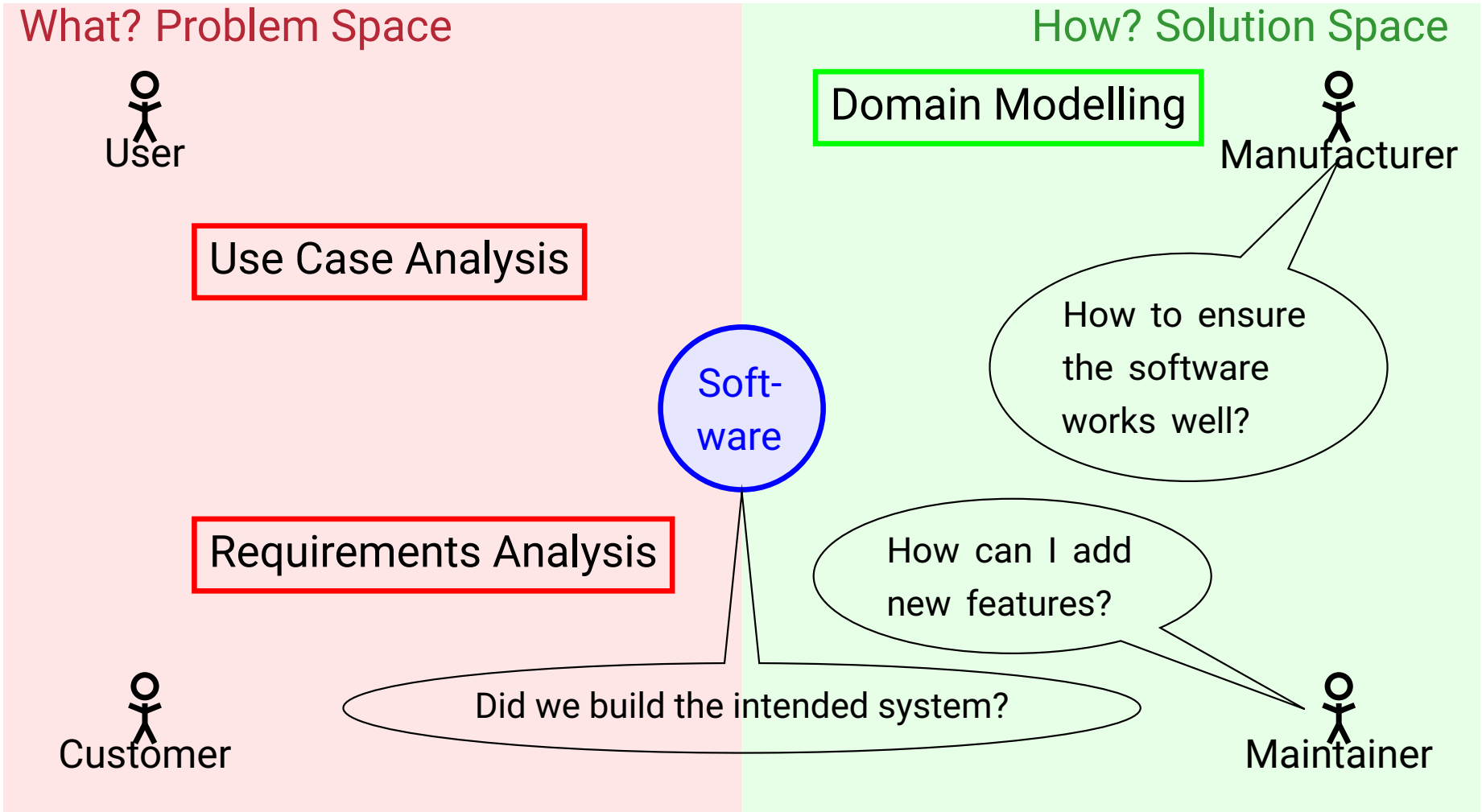
Software Engineering Group

What? vs How?


What? Problem Space
How? Solution Space
User
Manufacturer
How do I build the software?
Use Case Analysis
How to ensure the software works well?
Soft-ware
Requirements Analysis
How can I add new features?
Did we build the intended system?
Customer
Maintainer

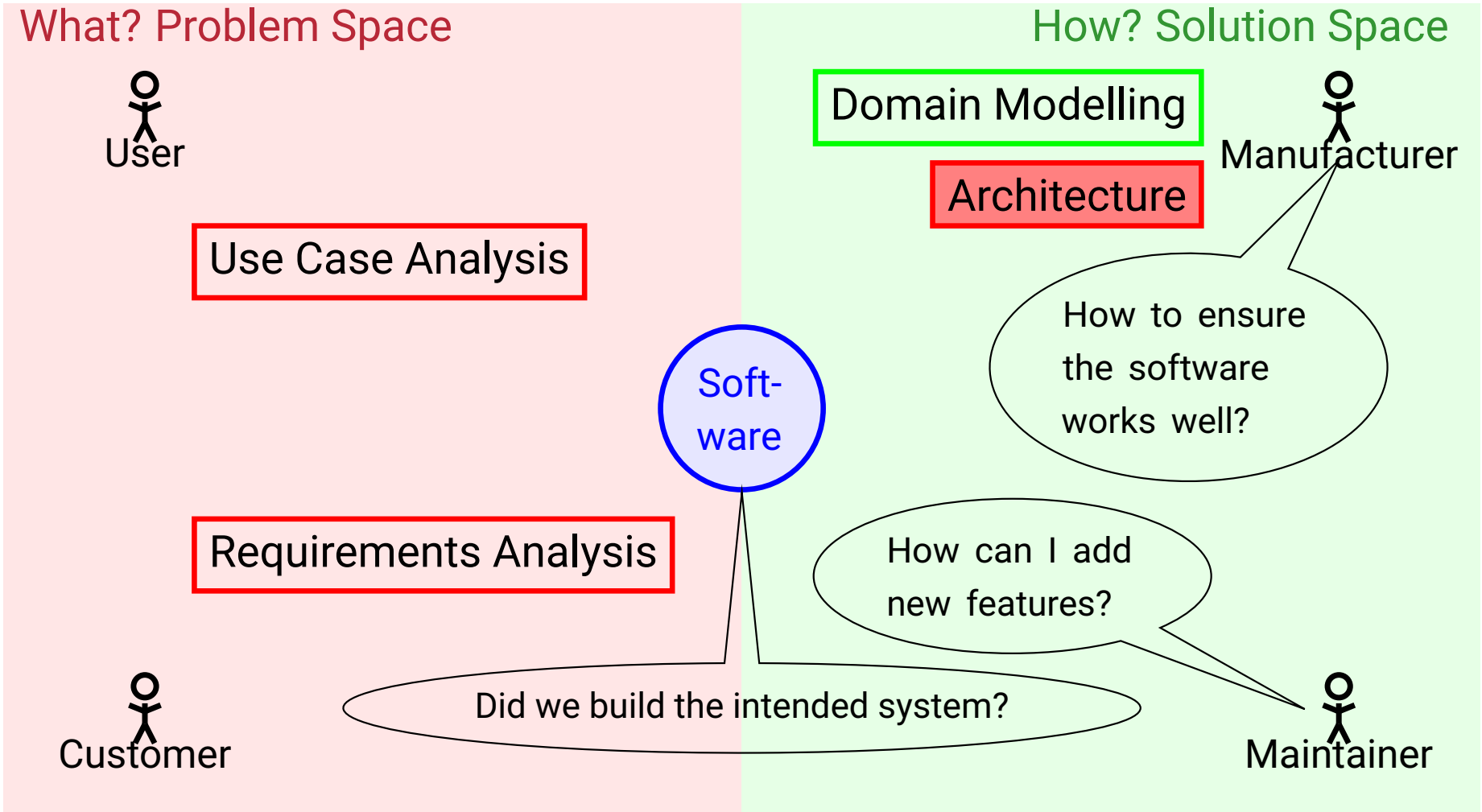
What? vs How?

TECHNISCHE UNIVERSITÄT DARMSTADT



Software Engineering Group

What? vs How?


What? Problem Space
How? Solution Space
User
Manufacturer
Domain Modelling
Architecture
Use Case Analysis
How to ensure the software works well?
Soft-ware
Requirements Analysis
How can I add new features?
Did we build the intended system?
Customer
Maintainer

Software Architecture
A Product of its Time

TECHNISCHE UNIVERSITÄT DARMSTADT

Software architecture is a quickly evolving area in software engineering

Software Architecture
A Product of its Time

TECHNISCHE UNIVERSITÄT DARMSTADT

Software architecture is a quickly evolving area in software engineering

Architectures favoured in the past are now discouraged

&

Architectures discouraged in the past are now preferred

Software Engineering Group

Software Architecture
A Product of its Time

TECHNISCHE UNIVERSITÄT DARMSTADT

Software architecture is a quickly evolving area in software engineering

Architectures favoured in the past are now discouraged

&

Architectures discouraged in the past are now preferred

Technology changes in the last 10 to 15 years that influenced architecture:

- Availability of
  - open interfaces to easily swap components such as databases
  - mature open source tools resulting in significant cost reduction
  - high-speed connectivity at low cost
- Container technology (lightweight virtualisation) and cloud services
  - containers can be easily added or removed as needed
  - made architectures like microservices feasible (scalability, elasticity)

Software Architecture

Dimensions that are part of it

TECHNISCHE UNIVERSITÄT DARMSTADT

Operational	Structural	Cross-Cutting
Availability	Extensibility	Accessibility
Scalability	Maintainability	Privacy
Performance	Leveragability	Security
...	...	...

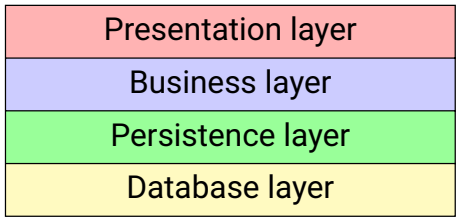
Software architecture encompasses ...

- Desired architecture characteristics

Software Engineering Group

Software Architecture

Dimensions that are part of it



Software architecture encompasses ...

- Desired architecture characteristics
- Architectural style (software system structure)

Software Architecture

Dimensions that are part of it

TECHNISCHE UNIVERSITÄT DARMSTADT

What kind of web framework shall be used?

What are each layer's responsibilities?

Which layers can communicate with each other?

What exchange data formats must be used?

Software architecture encompasses …

- Desired architecture characteristics
- Architectural style (software system structure)
- Decisions concerning architecture

Software Architecture

Dimensions that are part of it

TECHNISCHE UNIVERSITÄT DARMSTADT

NoSQL databases are the preferred storage option

Immutable data structures are preferred when applicable

Use asynchronous messaging between services when possible

Avoid usage of caching in clients whenever possible

Software architecture encompasses . . .

- Desired architecture characteristics
- Architectural style (software system structure)
- Decisions concerning architecture
- Design principles

Software Engineering Group

Architecture and Design

Demarcation line

TECHNISCHE UNIVERSITÄT DARMSTADT

Architects are responsible for . . .

- extracting the architecture characteristics from the requirement analysis
- choosing styles for the software system
- creating the component structure
  (concise & coherent parts of a system, above class level)

Software Engineering Group

Architecture and Design

Demarcation line

TECHNISCHE UNIVERSITÄT DARMSTADT

Architects are responsible for …

- extracting the architecture characteristics from the requirement analysis
- choosing styles for the software system
- creating the component structure
  (concise & coherent parts of a system, above class level)

input

Development team is responsible for …

- designing the class structure for each component (UML class diagrams)
  (class structure must meet architectural requirements)
- designing the user interface
- writing and testing the source code

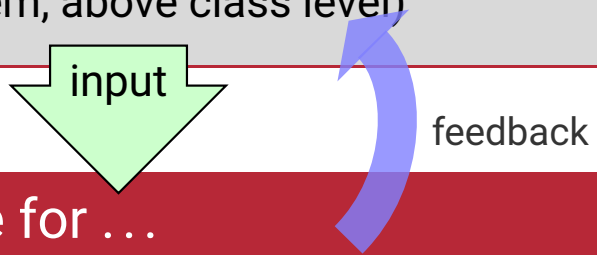
Architecture and Design

Demarcation line

TECHNISCHE UNIVERSITÄT DARMSTADT

Architects are responsible for …

- extracting the architecture characteristics from the requirement analysis
- choosing styles for the software system
- creating the component structure
  (concise & coherent parts of a system, above class level)



Development team is responsible for …

- designing the class structure for each component (UML class diagrams)
  (class structure must meet architectural requirements)
- designing the user interface
- writing and testing the source code

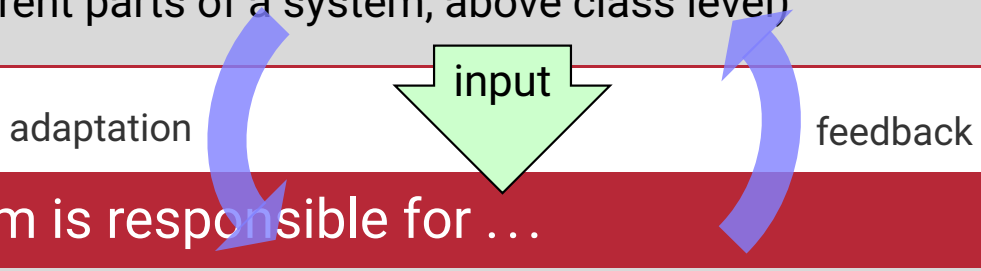
Architecture and Design

Demarcation line

TECHNISCHE UNIVERSITÄT DARMSTADT

Architects are responsible for …

- extracting the architecture characteristics from the requirement analysis
- choosing styles for the software system
- creating the component structure
  (concise & coherent parts of a system, above class level)



Development team is responsible for …

- designing the class structure for each component (UML class diagrams)
  (class structure must meet architectural requirements)
- designing the user interface
- writing and testing the source code

Software Engineering Group

Architecture and Design

Demarcation line

TECHNISCHE UNIVERSITÄT DARMSTADT

Architects are responsible for …

- extracting the architecture characteristics from the requirement analysis
- choosing styles for the software system
- creating the component structure
  (concise & coherent parts of

Modern architecture is iterative
and has feedback loop
Supports software evolution

adapt … edback

Development team is r … g for …

- designing the class structure for each component (UML class diagrams)
  (class structure must meet architectural requirements)
- designing the user interface
- writing and testing the source code

Software Engineering Group

Software Architecture

Dimensions that are part of it

TECHNISCHE UNIVERSITÄT DARMSTADT

Operational	Structural	Cross-Cutting
Availability	Extensibility	Accessibility
Scalability	Maintainability	Privacy
Performance	Leveragability	Security
...	...	...

Software architecture encompasses ...

- Desired architecture characteristics

Software Engineering Group

What is an Architecture Characteristic?

TECHNISCHE UNIVERSITÄT DARMSTADT

Criteria of an Architecture Characteristic

- Specifies a solution space design consideration:

  Specifies operational and design criteria concerning how to implement a given requirement
- Influences a structural design aspect:

  Requires the introduction of specific architectural elements (modules, components, . . .), not merely best practices in design/coding
- Critical/important that application performs as intended:

  Meets functional and non-functional requirements

Whether a specific characteristic meets the above defined criteria may depend on the context of the software system

Specific Architecture Characteristics Operational, Structural, Cross-Cutting

TECHNISCHE UNIVERSITÄT DARMSTADT

Operational

Availability	Times when system must be online/operational (when continuous, mechanism for quick failure recovery needed)
Performance	Encompasses peak analysis, response time, stress test, SLA
Scalability	Ability to function with increasing number of requests, users, …

Specific Architecture Characteristics Operational, Structural, Cross-Cutting

TECHNISCHE UNIVERSITÄT DARMSTADT

Structural

Extensibility	How easy is it to add new functionality?
Maintainability	Ease of adapting or enhancing the system
Leverageability	Reuse common components across multiple products
Localization	Languages, currencies, units, . . .
Configuration	End user can tailor system interface towards own needs

Specific Architecture Characteristics Operational, Structural, Cross-Cutting

TECHNISCHE UNIVERSITÄT DARMSTADT

Cross-Cutting

Accessibility Wide usability, barrier-free access

(diminished eyesight, hearing loss, reduced motion control)

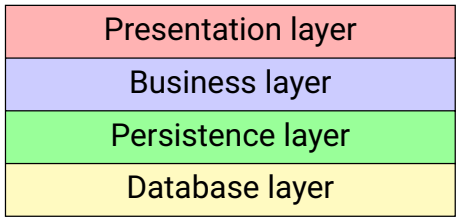
Privacy Ability to render data inaccessible for unauthorized users

(need to involve database or network architects)

Security Encryption of database, network traffic, end-to-end, etc.
authentication, authorization, resilience against attacks, . . .

Software Architecture

Dimensions that are part of it



Software architecture encompasses ...

- Desired architecture characteristics
- Architectural style (software system structure)

What do Architectural Styles Achieve?

TECHNISCHE UNIVERSITÄT DARMSTADT

Architectural styles . . .

- help to specify the fundamental structure of a software system
- impact the choice of concrete software architectures
- define a system's global properties such as:
  - how distributed components cooperate and exchange data
  - boundaries of subsystems

Different software system components may have different architecture style

Which Architectural Styles are there?

TECHNISCHE UNIVERSITÄT DARMSTADT

Representative architectural styles

- Layers
- Pipes and Filters
- Model-View-Controller

} monolithic

- Service-Based
- Broker
- Microservices

} distributed

Software Engineering Group

Which Architectural Styles are there?

TECHNISCHE UNIVERSITÄT DARMSTADT

Representative architectural styles

- Layers
- Pipes and Filters
- Model-View-Controller

} monolithic

- Service-Based
- Broker
- Microservices

} distributed

Software Engineering Group

Which Architectural Styles are there?

TECHNISCHE UNIVERSITÄT DARMSTADT

Representative architectural styles

- Layers
- Pipes and Filters
- Model-View-Controller

} monolithic

- Service-Based
- Broker
- Microservices

} distributed

Choosing an architectural style

Choice of architectural style (almost) always comes with trade-off.

It is a necessity to be aware of these, because additional measures might have to be taken to ensure that all architecture characteristics are met.

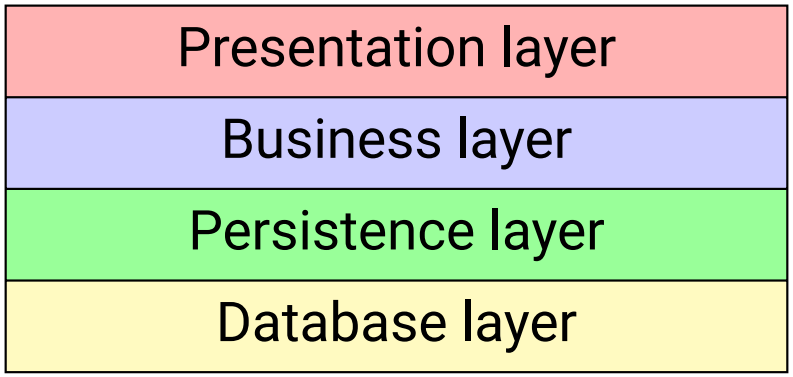
Part I

Monolithic Architectural Styles

Architectural Style: Layered

Topology

TECHNISCHE UNIVERSITÄT DARMSTADT



Topology

Number (and names) of layers not fixed

- Business and persistence layer can be unified
  (business objects create SQL queries themselves)
- Additional layers (e.g., service) to enable reuse or restrict access

Software Engineering Group

Software Architecture

Dimensions that are part of it

What kind of web framework shall be used?

What are each layer's responsibilities?

Which layers can communicate with each other?

What exchange data formats must be used?

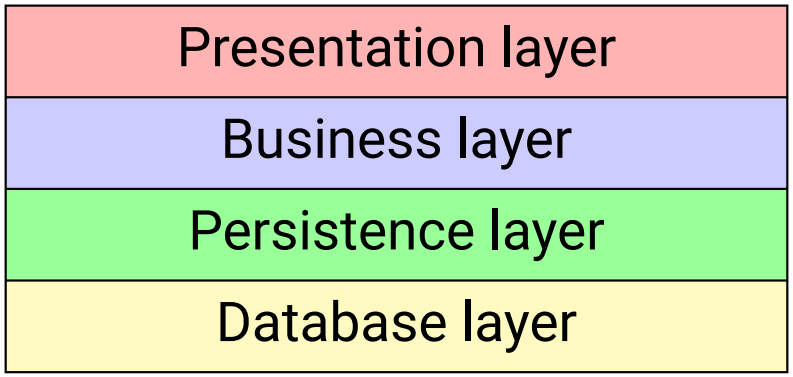
Software architecture encompasses …

- Desired architecture characteristics
- Architectural style (software system structure)
- Decisions concerning architecture

Architectural Style: Layered

Decisions

TECHNISCHE UNIVERSITÄT DARMSTADT

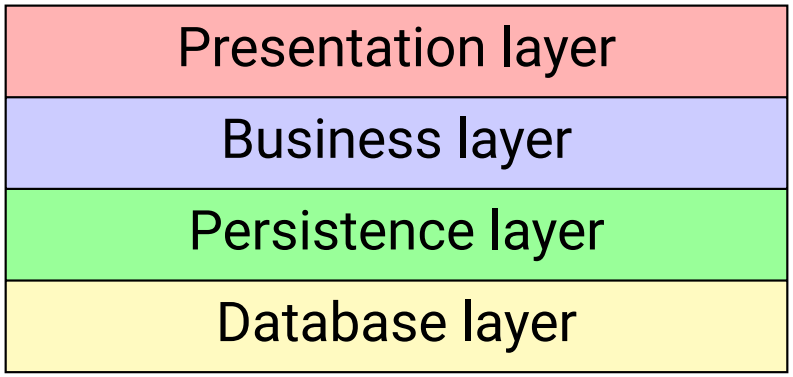


Decisions concerning architecture

- Definition of responsibilities for each layer
- Layer type (open or closed)
  - Top-Down request must pass through closed layers, can skip open layers
  - Open layers improve performance (less call chains), but lack encapsulation

Architectural Style: Layered

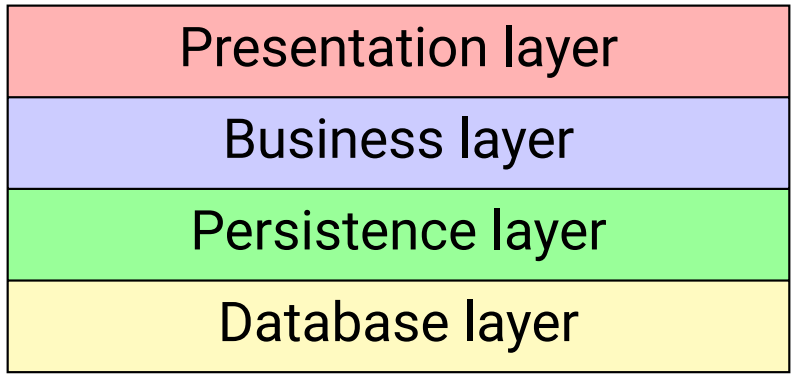
Trade-offs



Trade-Offs (in Selection)

+ Simplicity, Cost
+ Reliability (medium)
- Elasticity and scalability
- Performance
  (parallelization not inherently supported by architecture)
- Availability
  (often long startup time / long mean time to recovery)

Architectural Style: Layered
Summary

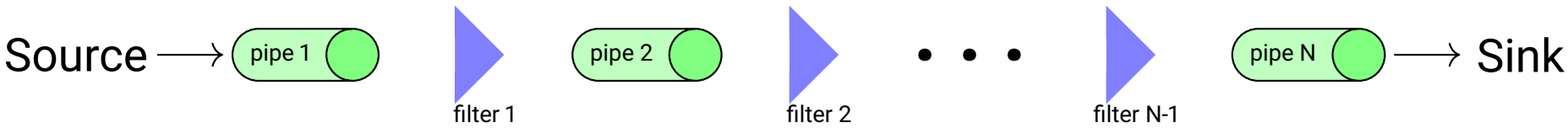


Summary

- Technological decomposition, not domain-driven
- Works well for small to medium-size applications
- May serve as initial architecture, to be changed later
- Problematic: Often not conscious decision, but result of Conway's law
  (Design reflects organisational structure of the manufacturer)

Architectural Style: Pipes and Filters

Topology



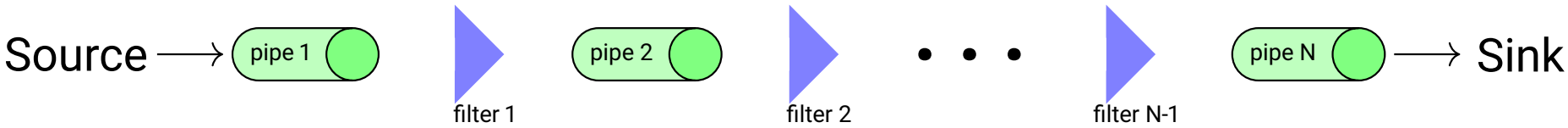
Main topology I

Pipes (“Röhren”, “Fliessband”)

- Uni-directional, point-to-point channel from a data source to a target
- Permit any data format (i.e. not restricted by this style)
- Formats supporting smaller data chunks are preferable for performance/throughput

Architectural Style: Pipes and Filters

Topology



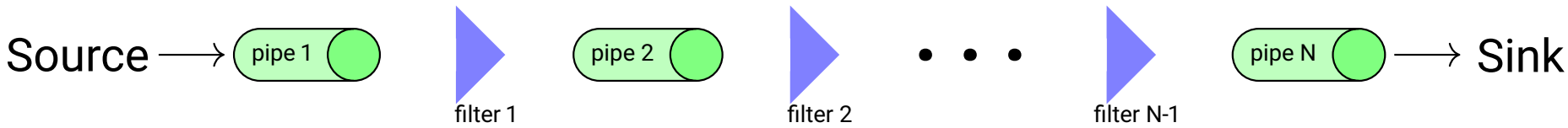
Main topology II

Filters

- Self-contained and independent (from other filters) units
- Stateless: behavior does not depend on past actions
- Realize exactly one task

Architectural Style: Pipes and Filters

Topology



Main topology III

Kinds of filters

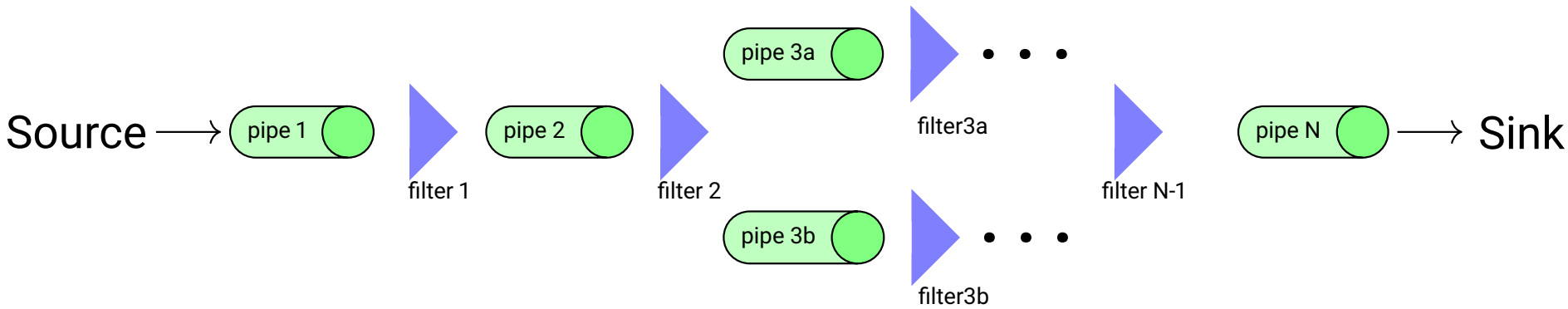
Producer: Data source

Transformer: (i) Receives data from its input channel(s)
(ii) Performs operations on the data
(iii) Forwards the result via output channel

Consumer: Data sink, output to display, file, or database

Architectural Style: Pipes and Filters

Variant



Variant Topology (Branching)

Additional kind of filter

Tester: (i) Receives input
(ii) Tests whether data satisfies given condition
(iii) Redirects data to output channel according to test outcome

Architectural Style: Pipe and Filters

Example & Usage

TECHNISCHE UNIVERSITÄT DARMSTADT

Example (Image processing)

Software Engineering Group

Architectural Style: Pipe and Filters

Example & Usage

TECHNISCHE UNIVERSITÄT DARMSTADT

Example (Image processing)

Pipes and Filters typically used as Sub-architecture

- Text, image and audio processing
- Streaming (Apache Kafka platform, . . .)
- Neural networks can be viewed as pipes & filters

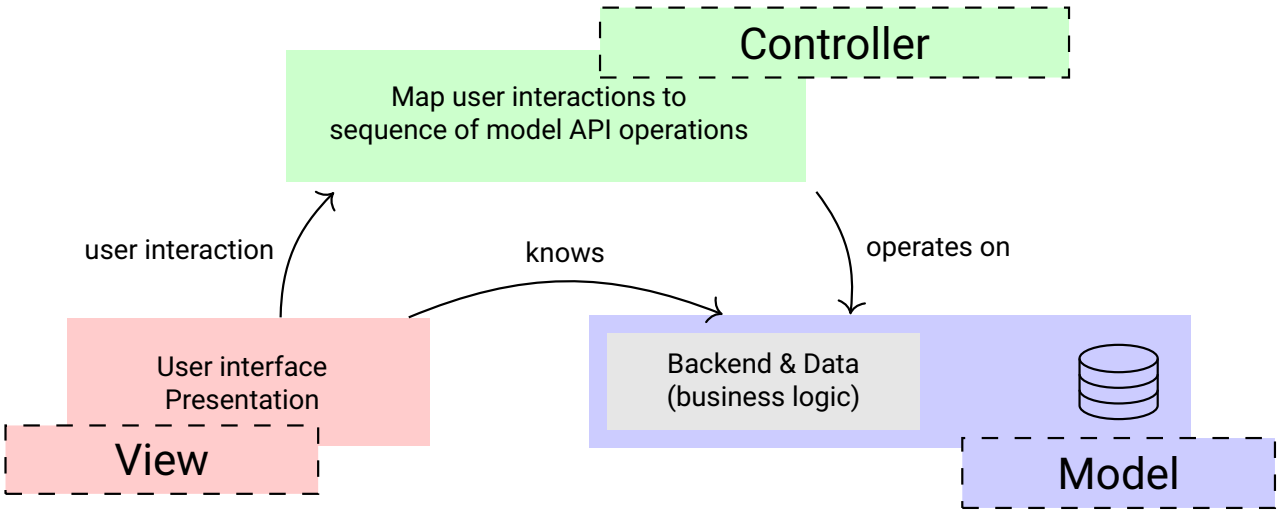
Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

Topology & Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

Fundamental structural organization for interactive software systems



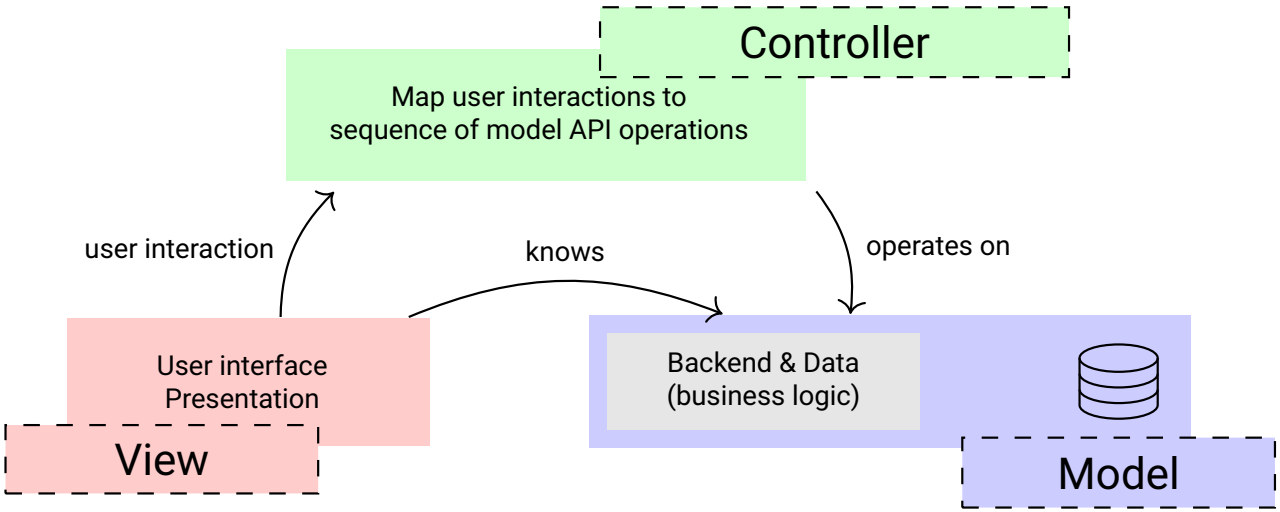
Topology

- Separates system in three modules: Model—View—Controller
  Domain-driven decomposition

Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

Topology & Structure



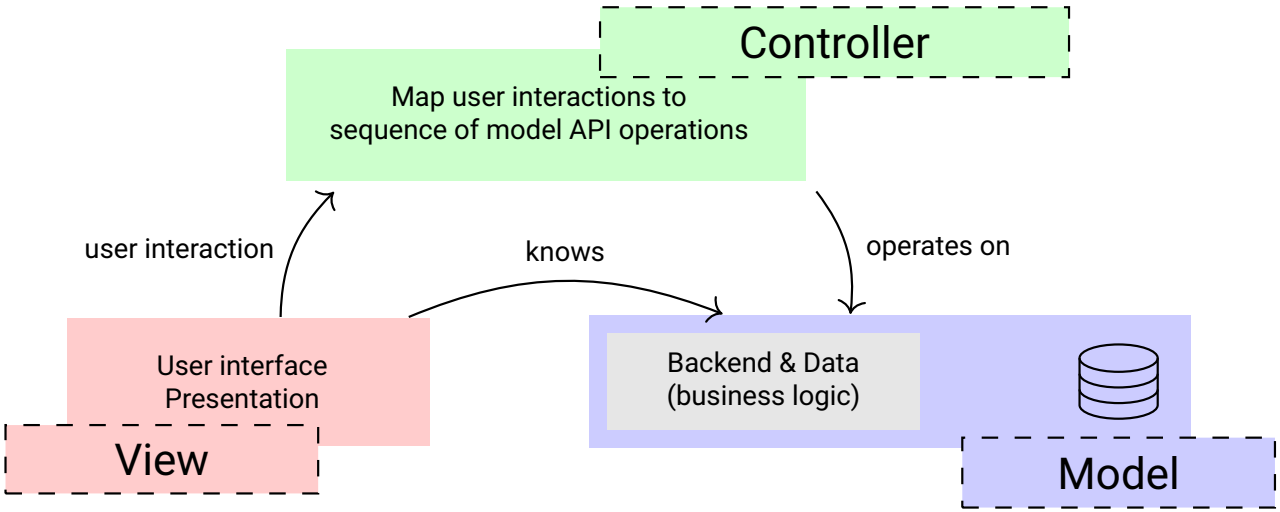
Topology

- Separates system in three modules: Model—View—Controller
  - Model: Business logic and data storage
    - No handling of interactions
    - Not concerned with presentation

Architectural Style: Model-View-Controller (MVC)

Topology & Structure

TECHNISCHE UNIVERSITÄT DARMSTADT



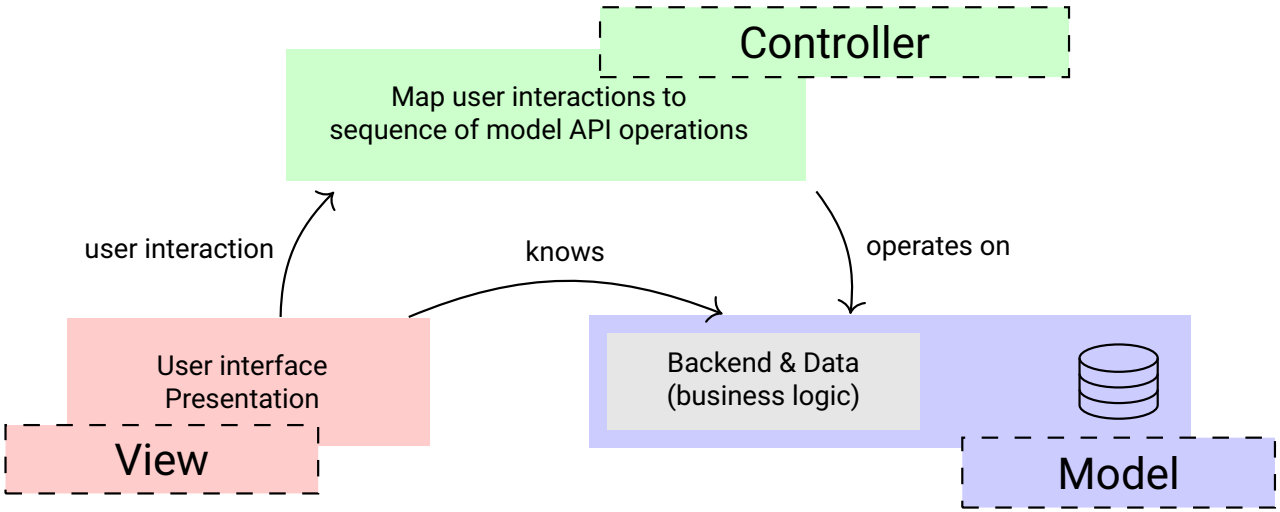
Topology

- Separates system in three modules: Model—View—Controller
  - Model: Business logic and data storage
  - View: Presentation or view of data to the user
    - Data obtained from model
    - Frequently, different views for different purposes

Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

Topology & Structure

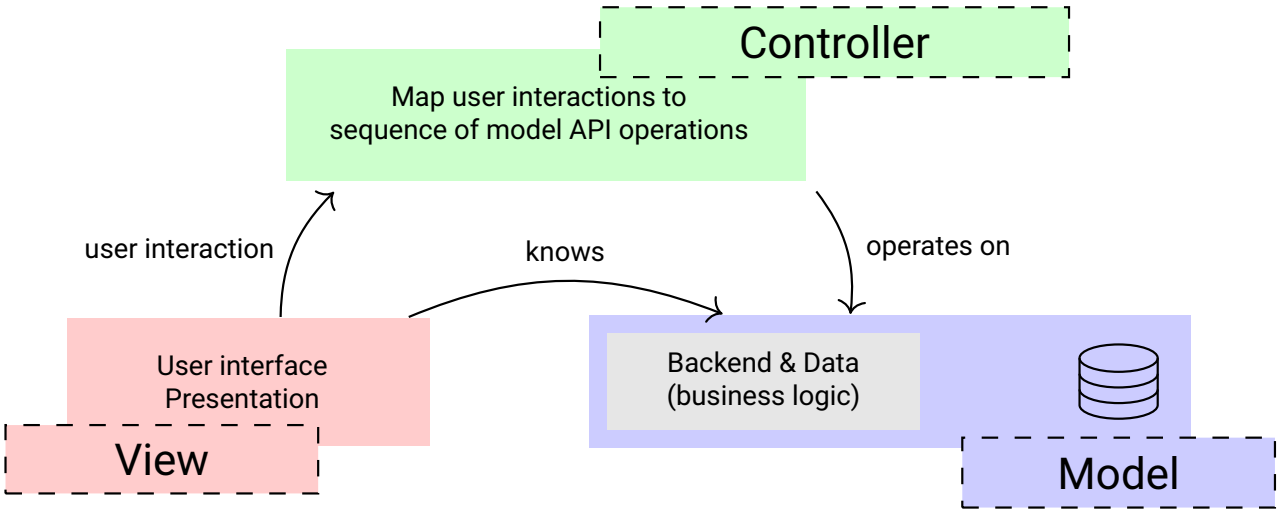


Topology

- Separates system in three modules: Model—View—Controller
  - Model: Business logic and data storage
  - View: Presentation or view of data to the user
  - Controller: Translation of user interaction to model operations
    - Each view associated to specific controller
    - All interactions done via a controller

Architectural Style: Model-View-Controller (MVC)

Topology & Structure

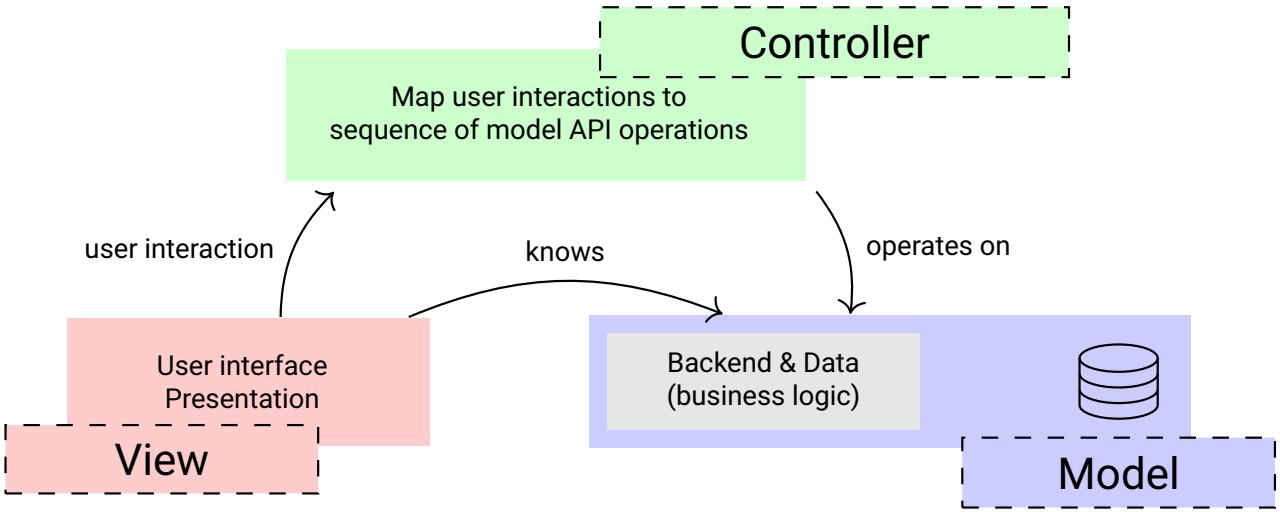


Topology

- Separates system in three modules: Model—View—Controller
- Controller and View are directly coupled with the Model

Architectural Style: Model-View-Controller (MVC)

Topology & Structure

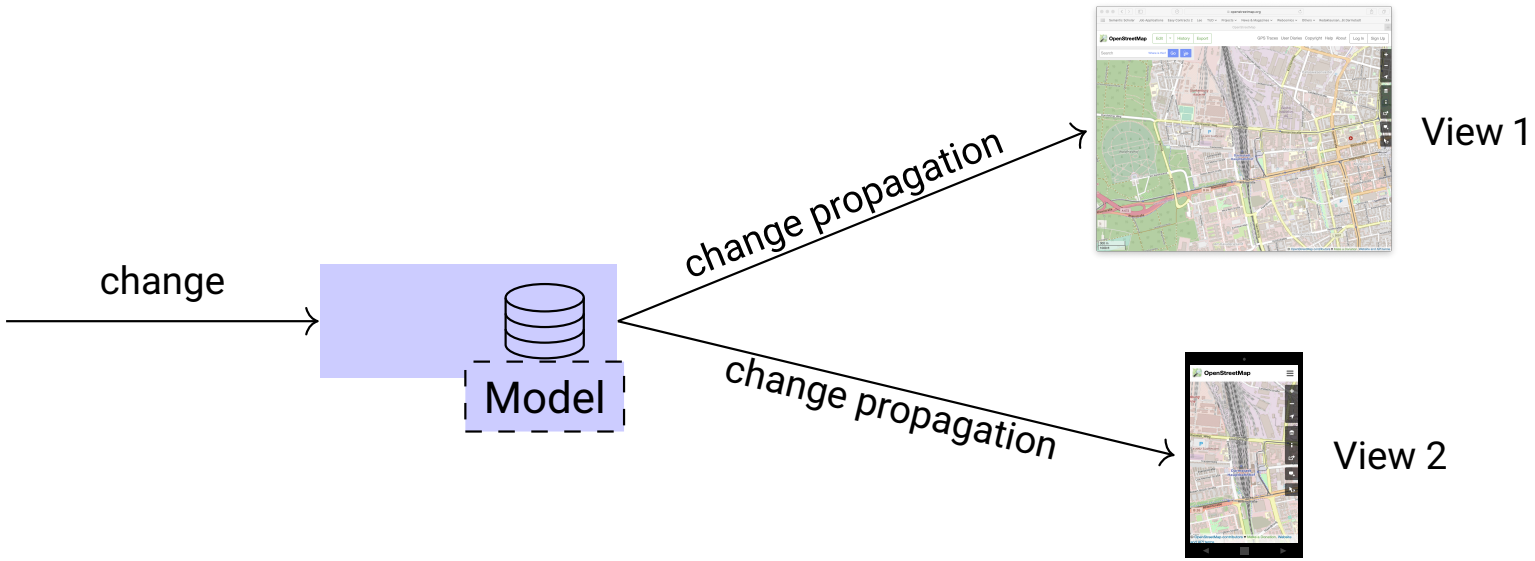


Topology

- Separates system in three modules: Model—View—Controller
- Controller and View are directly coupled with the Model
- Model is not directly coupled with the Controller or the View

Architectural Style: Model-View-Controller (MVC)

Change Propagation



Change Propagation

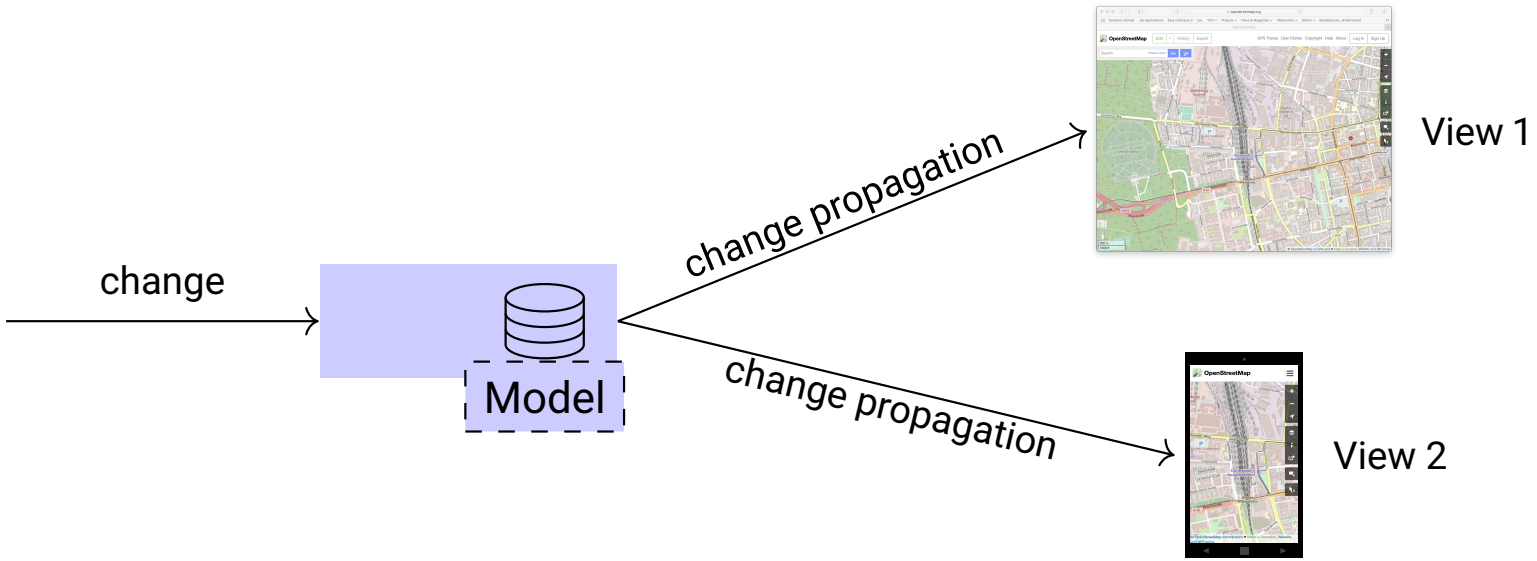
Change propagation mechanism ensures Model-View consistency

(User interface consistent with model data)

(Often implemented using the Observer/Publisher-Subscriber pattern: see later lecture)

Architectural Style: Model-View-Controller (MVC)

Change Propagation

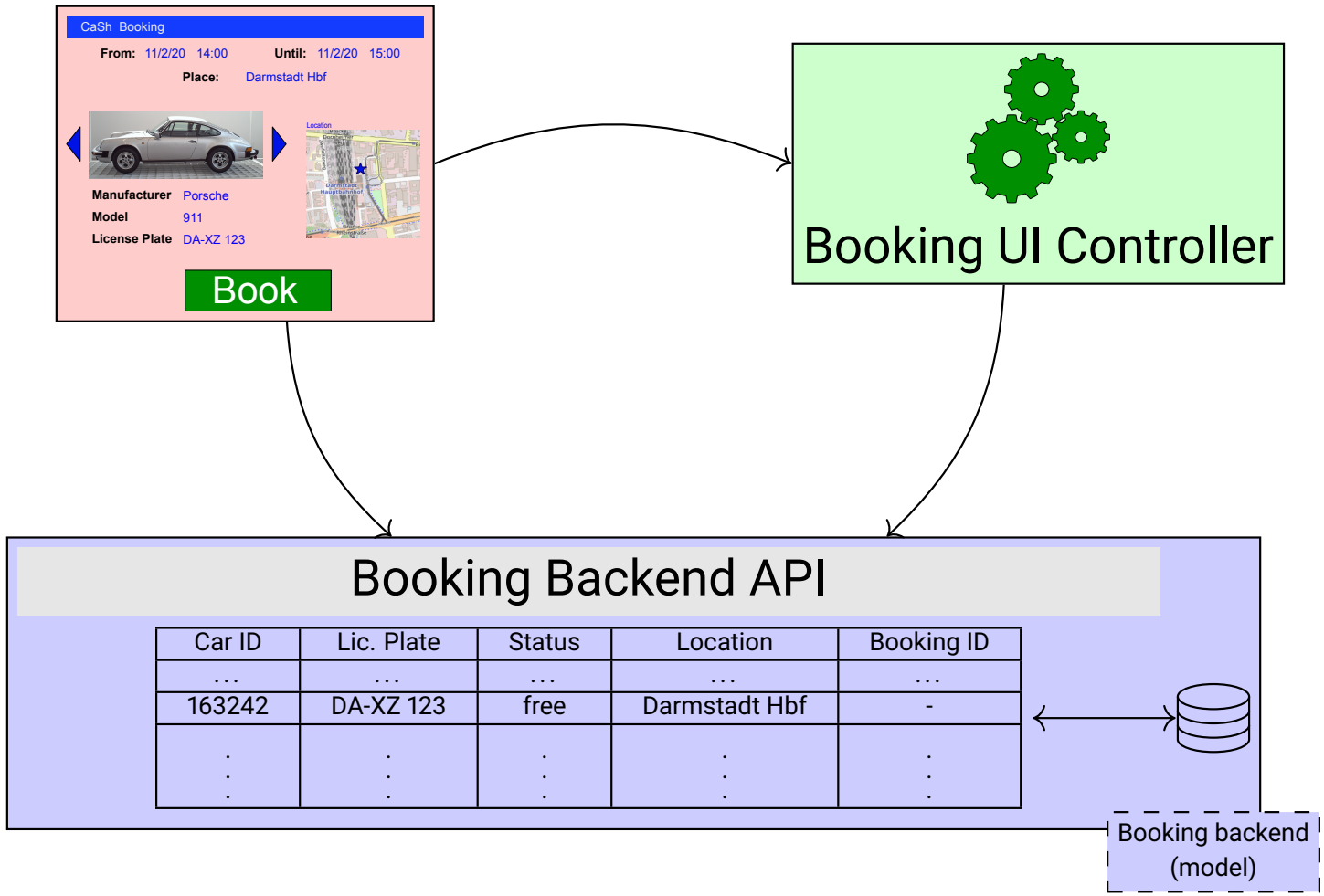


Change Propagation Implementation Principle

- Views register themselves at a model
  (Likewise, Controllers, whose behavior depends on the model state)
- Model notifies registered view objects upon change

Architectural Style: Model-View-Controller (MVC)

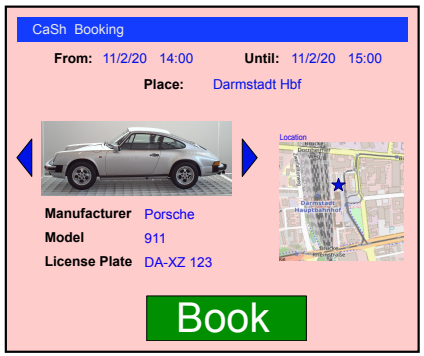
CaSh Example: Static Structure

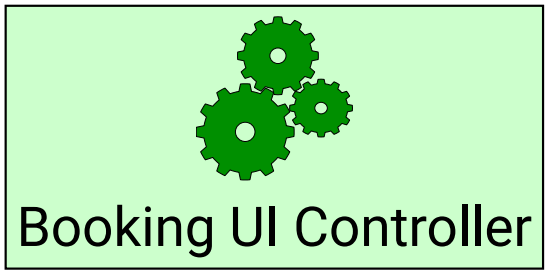


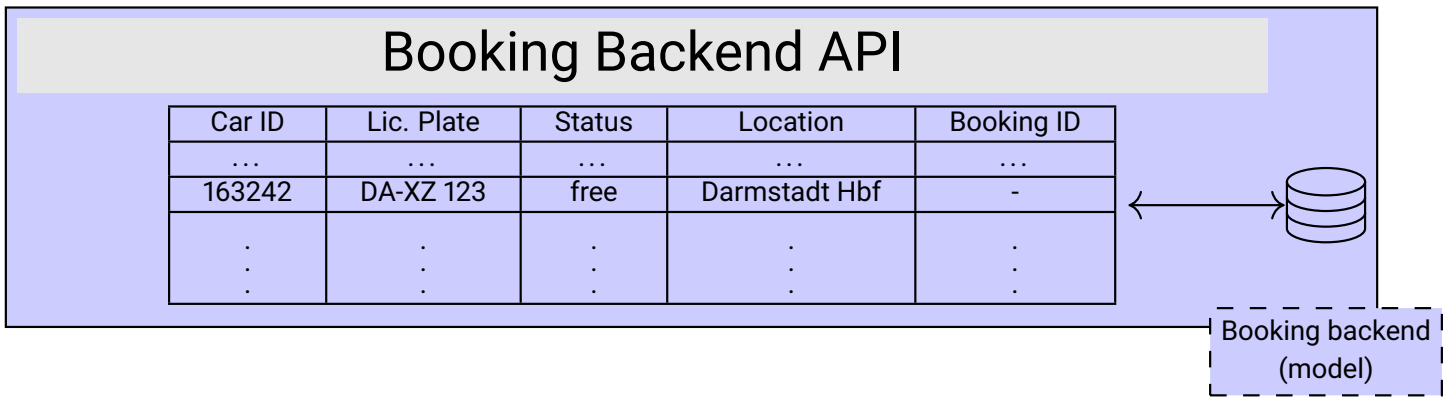
Architectural Style: Model-View-Controller (MVC)

CaSh Example: Dynamic Behavior

TECHNISCHE UNIVERSITÄT DARMSTADT





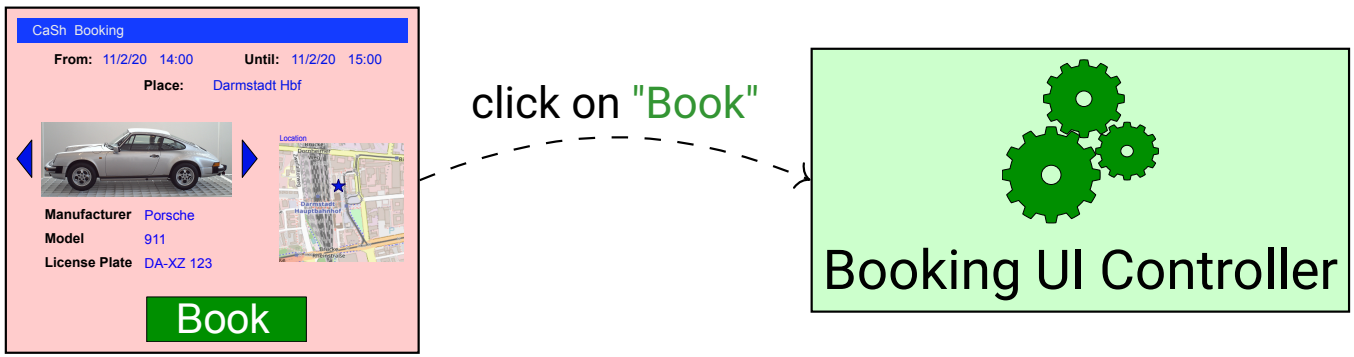


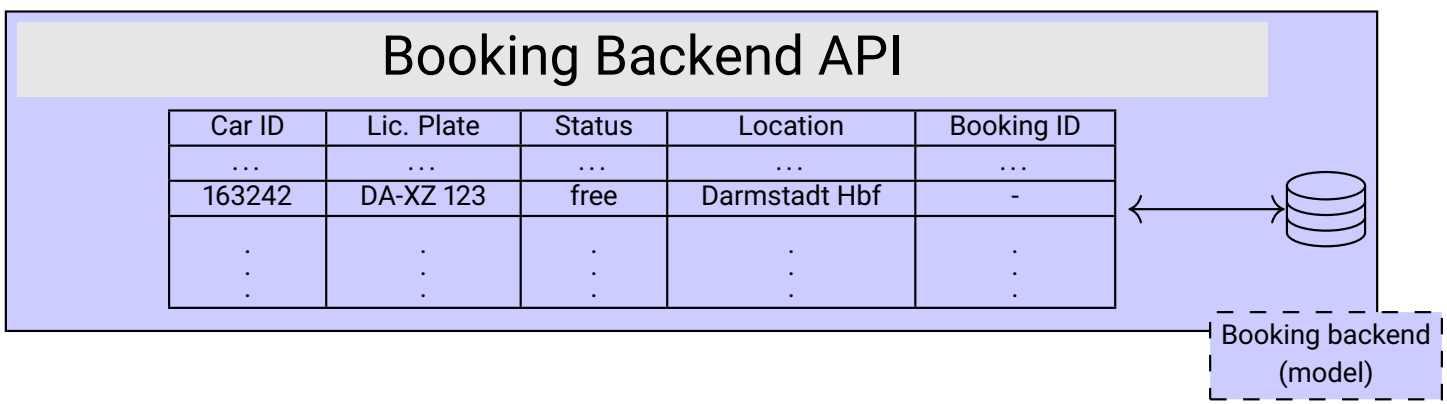
Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

CaSh Example: Dynamic Behavior

TECHNISCHE UNIVERSITÄT DARMSTADT



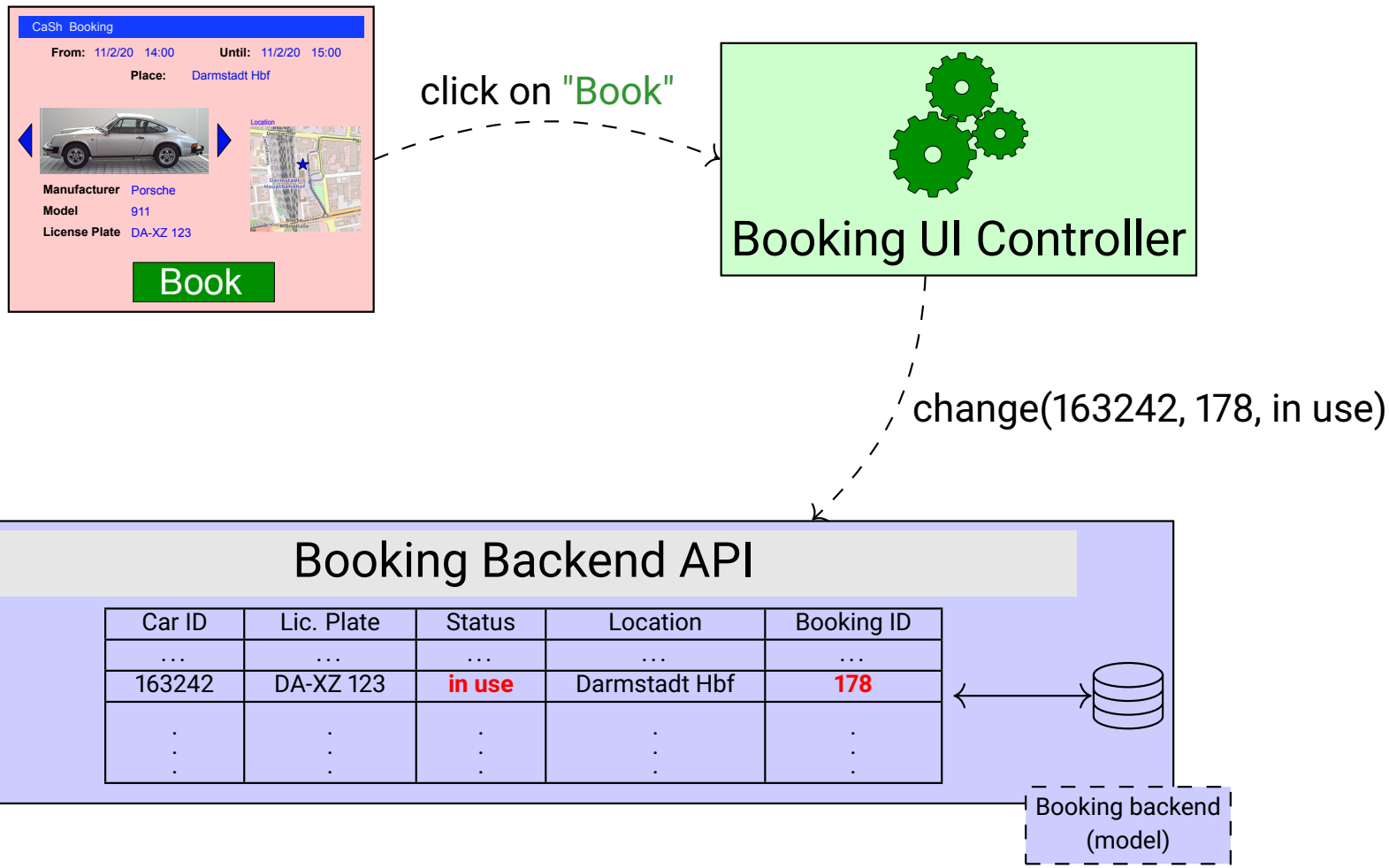


Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

CaSh Example: Dynamic Behavior

TECHNISCHE UNIVERSITÄT DARMSTADT

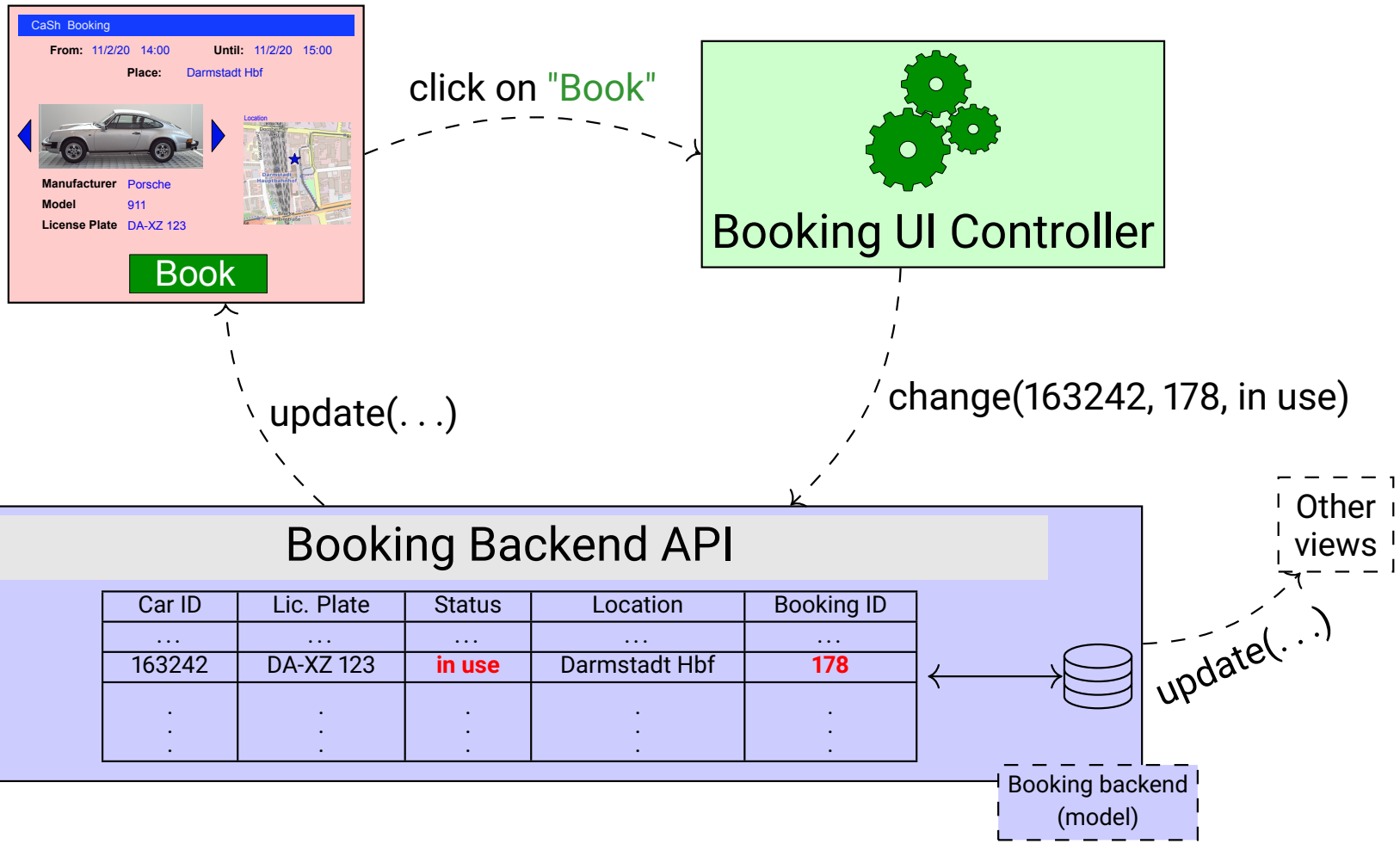


Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

CaSh Example: Dynamic Behavior

TECHNISCHE UNIVERSITÄT DARMSTADT



Software Engineering Group

Architectural Style: Model-View-Controller (MVC)

Applicability and Trade-Offs

TECHNISCHE UNIVERSITÄT DARMSTADT

When to use MVC

Application is interactive and:

- Model data to be presented in different views
- Number and kind of views not fixed (or unknown)
- Display and application behavior must reflect data changes immediately
- Changing UI should not affect the business logic and data storage

Booking system, information system, information panels (airport, train)

Architectural Style: Model-View-Controller (MVC)

Applicability and Trade-Offs

TECHNISCHE UNIVERSITÄT DARMSTADT

When to use MVC

Application is interactive and:

- Model data to be presented in different views
- Number and kind of views not fixed (or unknown)
- Display and application behavior must reflect data changes immediately
- Changing UI should not affect the business logic and data storage

Trade-Offs / Caveats

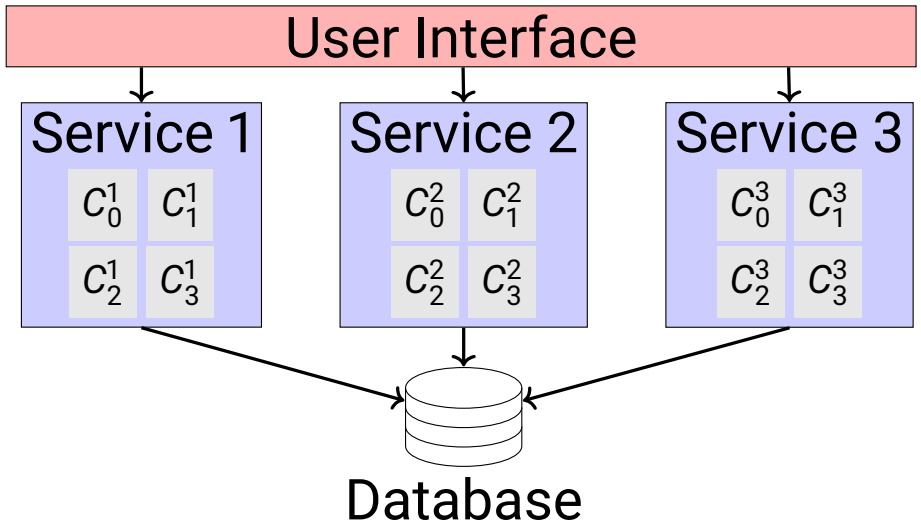
1. Update proliferation, but not all views are interested in all changes
2. Increase in complexity due to separate view and controller components without gaining much flexibility
3. High dependency (coupling) between view and controller

Part II

Distributed Architectural Styles

Architectural Style: Service-Based

Topology

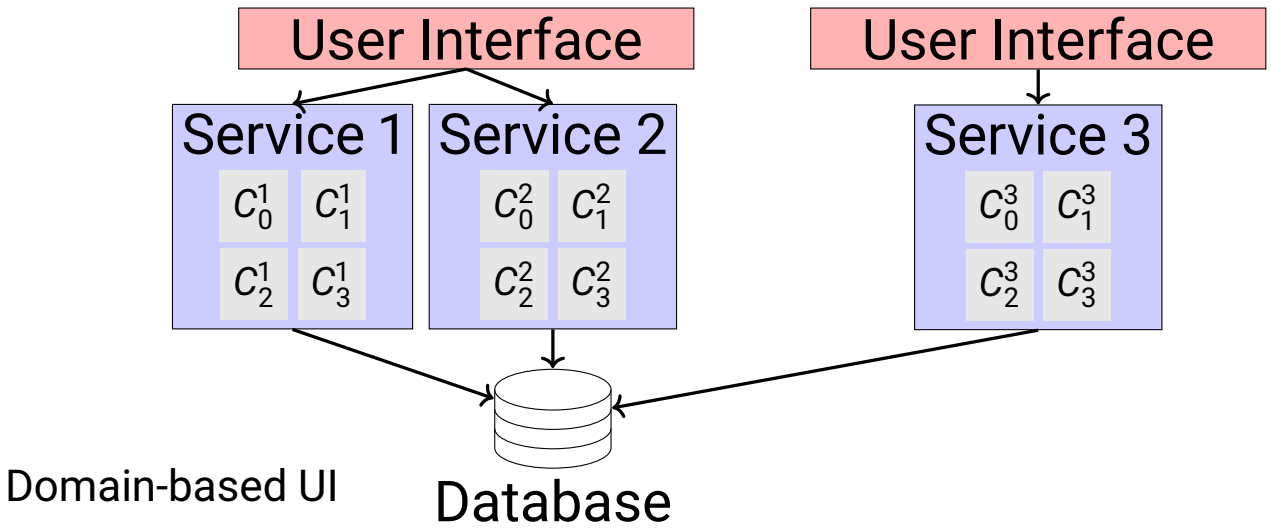


Topology

- One monolithic user interface for different types of services
- Each service *i* is divided into different components C_j^i
- Services share a common database

Architectural Style: Service-Based

Topology Variants

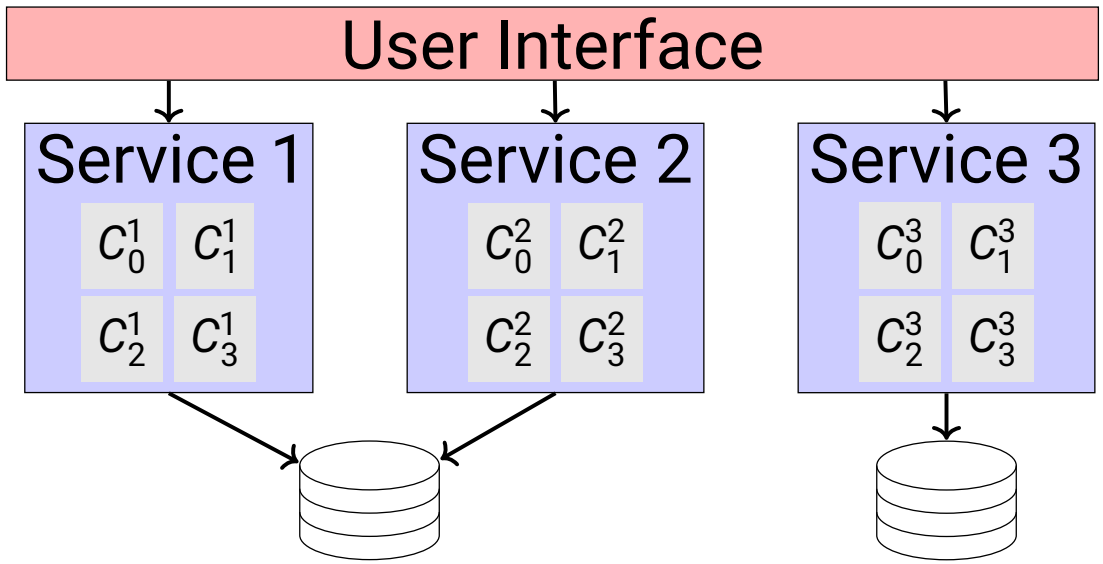


Non-monolithic user interfaces

- Domain-based user interface
- Service-based user interface (one UI per service; not shown)

Architectural Style: Service-Based

Topology Variants



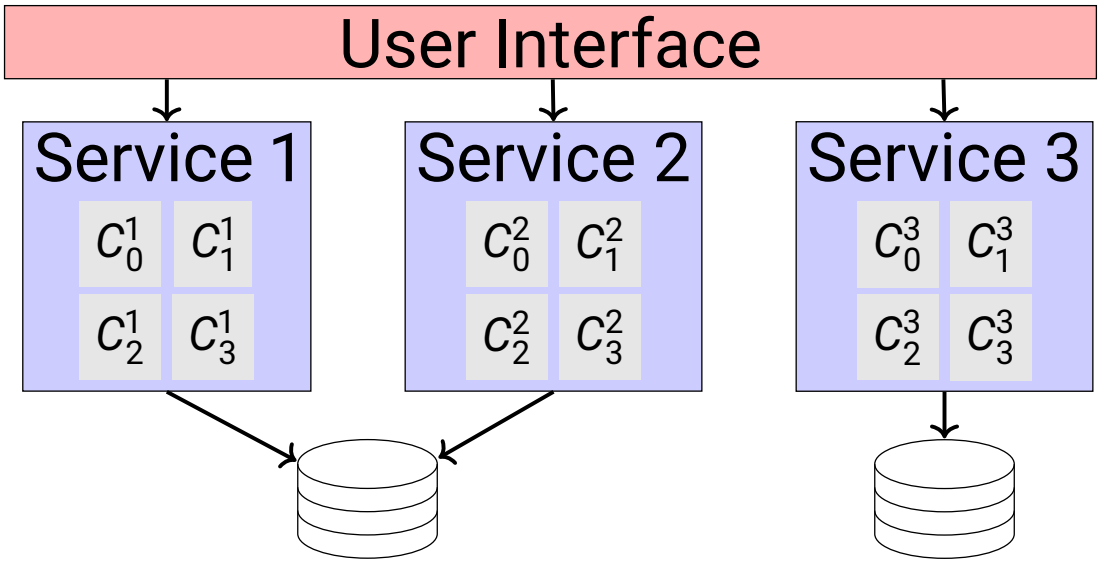
Distributed Databases

- Service-local database

Architectural Style: Service-Based

Topology Variants

TECHNISCHE UNIVERSITÄT DARMSTADT



Distributed Databases

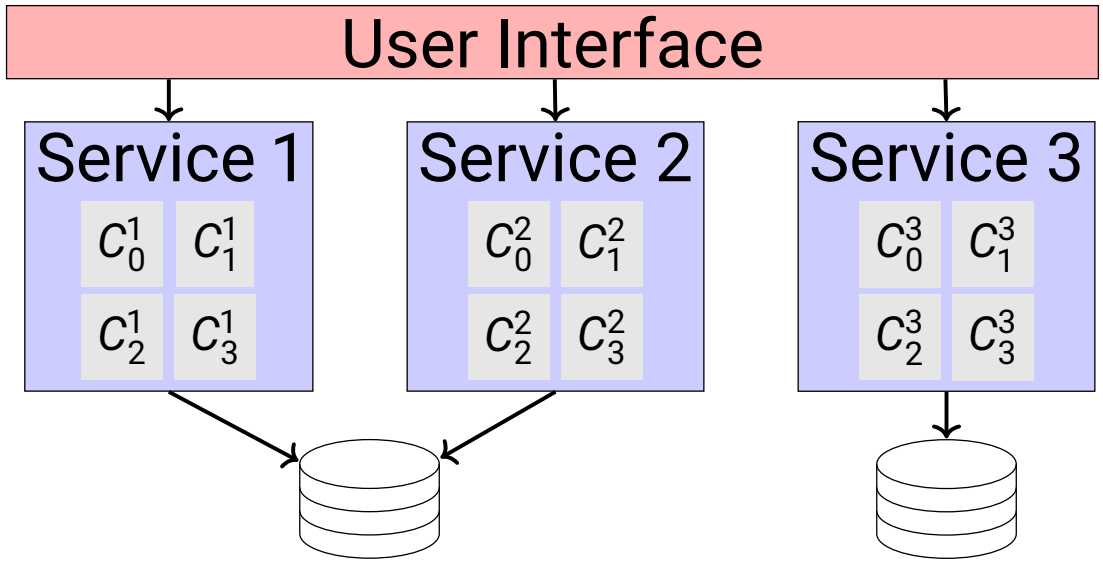
- Service-local database (shown on right)
- Data owned by one service (type)
- Topology helps data consistency and scalability

Software Engineering Group

Architectural Style: Service-Based

Topology Variants

TECHNISCHE UNIVERSITÄT DARMSTADT



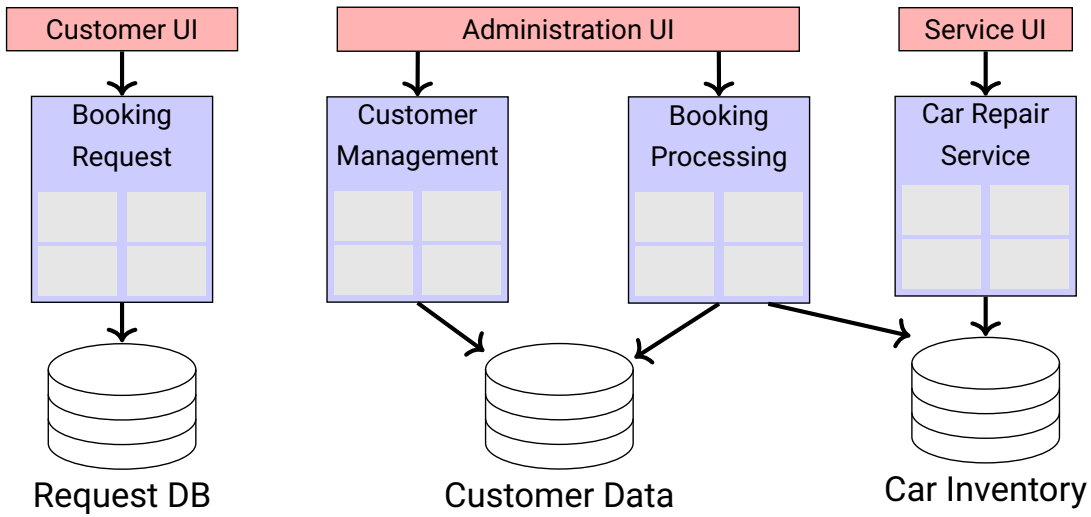
Distributed Databases

- Service-local database mixed with shared database (on left)
- Shared databases require coordination (in particular, shared schemata)
- Targeted data ownership: one service has write permissions, others read

Software Engineering Group

Architectural Style: Service-Based

CaSh Example



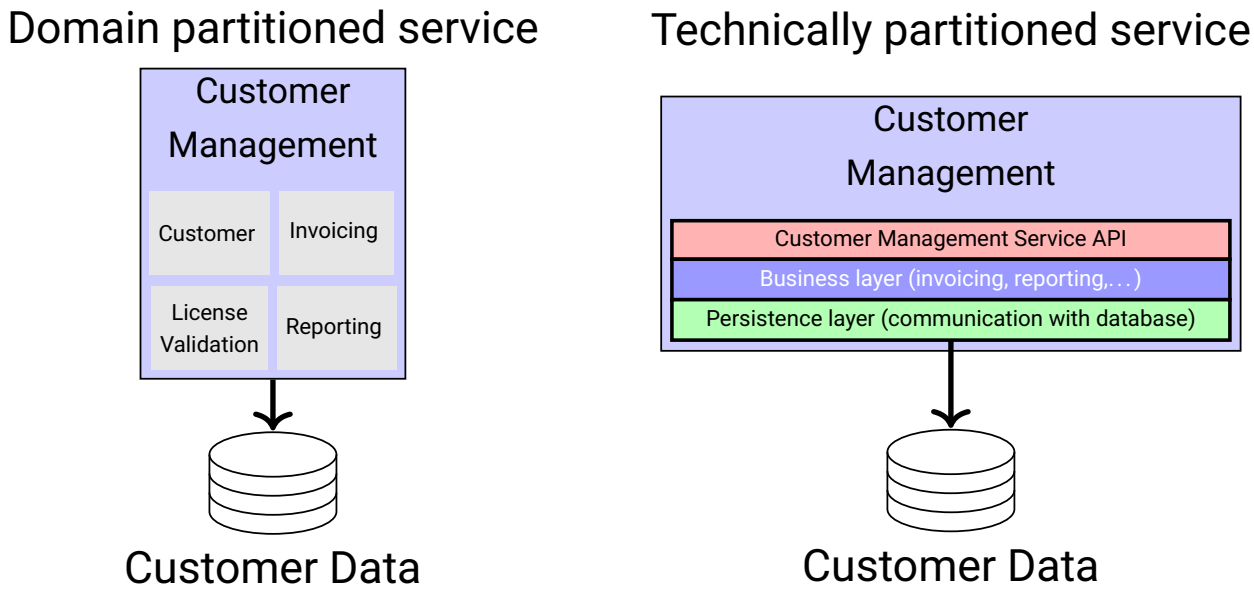
A service-based architecture for CaSh

- Customer UI uses Booking Request service to generate a booking request
- Booking request delegated to Booking Processing service
- Customer services have no direct access to company databases (architecture supports security aspects)

Architectural Style: Service-Based

CaSh: Service Structure

TECHNISCHE UNIVERSITÄT DARMSTADT



CaSh: Alternative Service Architectures

- Domain partitioned service:
  components realize domain-specific tasks
- Technically partitioned service:
  components organized as layered structure

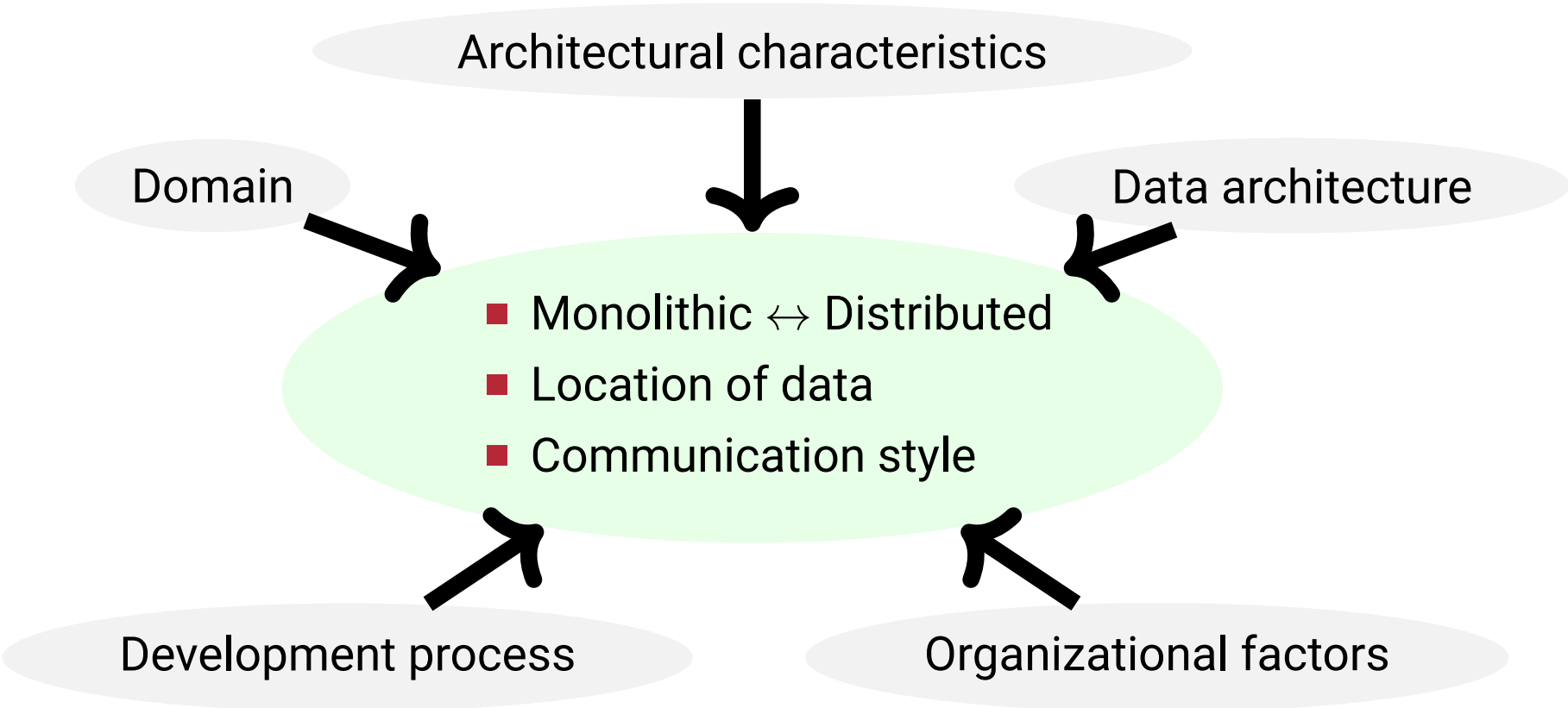
Software Engineering Group

Part III

Wrapping Up

Choosing an Architectural Style

Influences



Software Architecture

Dimensions that are part of it

TECHNISCHE UNIVERSITÄT DARMSTADT

What kind of web framework shall be used?

What are each layer's responsibilities?

Which layers can communicate with each other?

What exchange data formats must be used?

Software architecture encompasses . . .

- Desired architecture characteristics
- Architectural style (software system structure)
- Decisions concerning architecture

Software Engineering Group

Decisions Concerning Architecture

Soft Skill Aspects

TECHNISCHE UNIVERSITÄT DARMSTADT

Fear of the Wrong Decision

Decision is avoided/postponed for fear to make the wrong choice

- Ok to make decision at “last responsible moment”
  (Early enough so that the lacking decision does not delay other work)
- Stay close to development team to be able to react to problems

Software Engineering Group

Decisions Concerning Architecture

Soft Skill Aspects

TECHNISCHE UNIVERSITÄT DARMSTADT

Infinite discussions

Decision discussed over and over again because its reason is not known

Decisions Concerning Architecture

Soft Skill Aspects

TECHNISCHE UNIVERSITÄT DARMSTADT

Infinite discussions

Decision discussed over and over again because its reason is not known

Architecture not implemented

Developers are unaware of architectural decisions

Software Engineering Group

Decisions Concerning Architecture

Soft Skill Aspects

TECHNISCHE UNIVERSITÄT DARMSTADT

Infinite discussions

Decision discussed over and over again because its reason is not known

Architecture not implemented

Developers are unaware of architectural decisions

Both problems root in documentation and communication

- Document decisions at generally known and suitable place
  - Formal document, better: Wiki with issue tracker for development team
- Choose appropriate means to communicate (changes to) decisions
  - Use email, slack, . . . for communication (not only documentation)
  - Keep messages concise, refer to wiki/issue tracker/document for details

Difficulty to arrive at decision may point to gap in analysis phases

Software Engineering Group

Literature

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ian Sommerville: Software Engineering, 10th edition, Chapter 6, Pearson Education, 2015

  TUDa ULB eBook (German edition)
- Mark Richards and Neal Ford: Fundamentals of Software Architecture: An engineering approach, 1st edition, O'Reilly, 2020