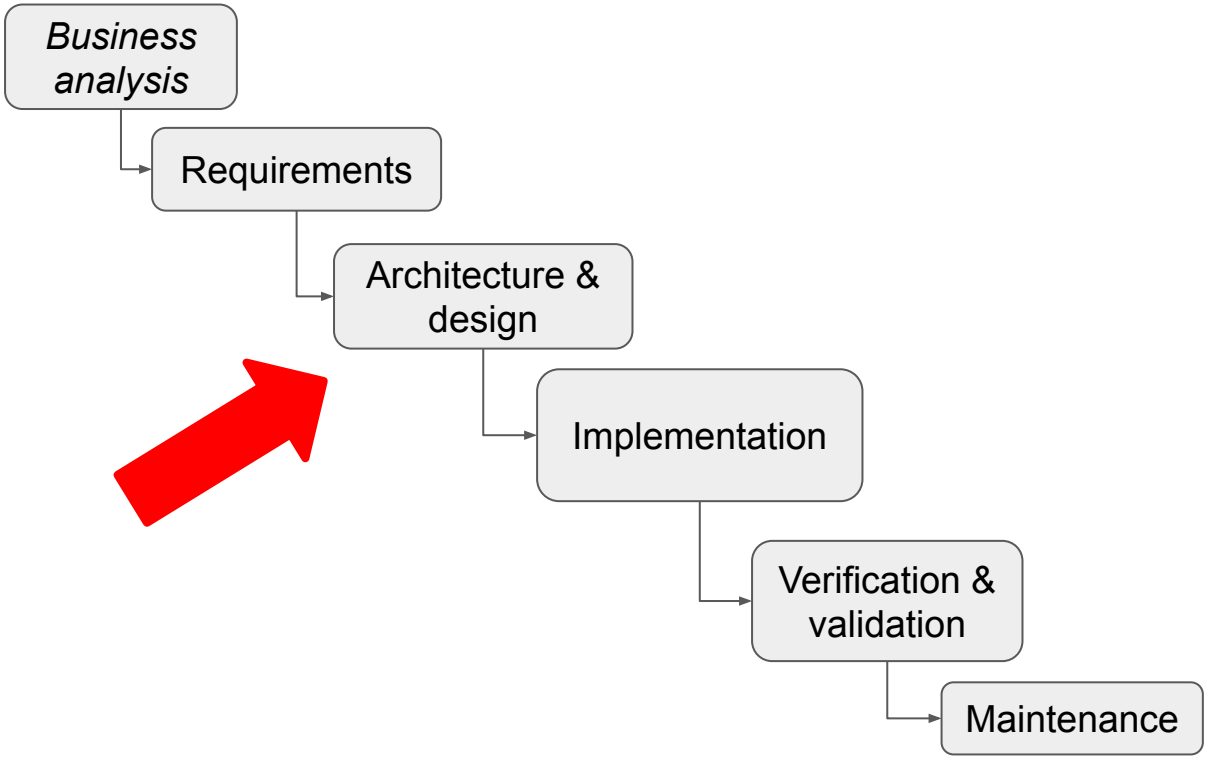
Architecture & design

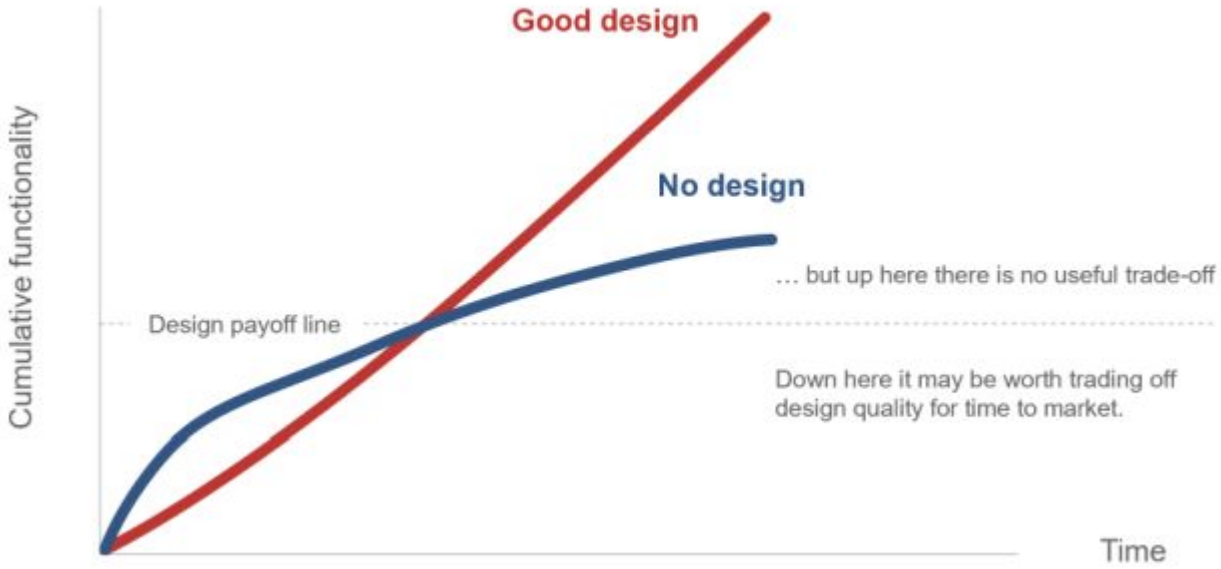
Jana Kostičová, 16.10.2024

SDLC - Waterfall



- Camelot is based on the **client-server model** and uses **remote procedure calls** both locally and remotely to provide communication among applications and servers."
- "**Abstraction layering** and system decomposition provide the appearance of system uniformity to clients, yet allow Helix to accommodate a diversity of autonomous devices. The architecture encourages a **client server model** for the structuring of applications."
- "We have chosen a **distributed, object-oriented approach** to managing information."
- "The easiest way to make the canonical sequential compiler into a concurrent compiler is to **pipeline** the execution of the compiler phases over a number of processors. . . . A more effective way [is to] split the source code into many segments, which are concurrently processed through the various phases of compilation [by multiple compiler processes] before a final, merging pass recombines the object code into a single program."

Is it worth the effort to design software well? [4]



Software architecture

= the organization or structure of a system, where the system represents a collection of components that accomplish a specific function or set of functions.

- A (software) component is a part of a software system that encapsulates a specific piece of functionality, e.g., libraries, modules, services, web components, plugins, ..
- Components serve as the building blocks for the structure of a system
- Components are connected via interfaces
- Components are typically specified in different views to show the relevant functional and non-functional properties of a software system

Interface

(Software) interface is a shared boundary across which two or more separate (software) components of a computer system exchange information.

- ABI - Application binary interface - typically not relevant (created by compiler / other tools).
- API - Application programming interface
- User interfaces

Architecture vs design

Software architecture

- High-level structure of the entire system and its division into a set of components

Software design

- Internal structure of individual components

4+1 architectural view model

= a model "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views" [6]

1. Logical view
2. Process view
3. Development view
4. Physical view

\+ scenarios

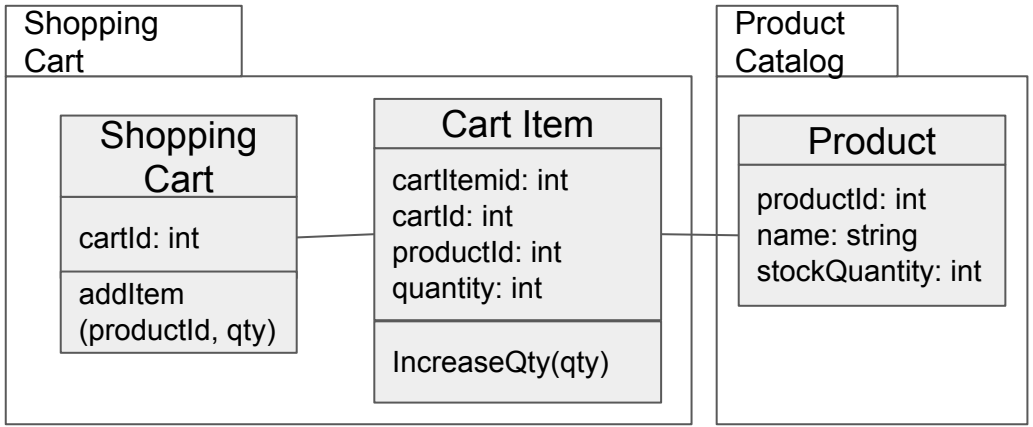
4+1 architectural view model: Logical view

- Describes the system in terms of components, their relations, and the functionality they provide
- The perspective of end users (and stakeholders in general)
- Overlap with “Requirements” phase
- *UML: Use Case diagrams, UML Class diagrams, …*

Example:

E-commerce store may contain classes:

- Shopping Cart
- Cart Item
- Product
- …



4+1 architectural view model: Process view

- Captures dynamic behavior of the system - the interactions and collaborations among processes, tasks, threads, and components during runtime
- Important for understanding concurrency, performance, and resource utilization.
- *UML: Sequence diagram, Communication diagram, Activity diagram, …*

Example:

E-commerce store may contain processes:

- AddItemProcess: Handles adding item to a shopping basket
- CheckoutProcess: Manages completion of an order

4+1 architectural view model: Development view

- Describes software organization - SW modules and components, their relationships, source code organization, ….
- Mapping of components from the Logical view into implementation
- The perspective of developers
- *UML: Package diagrams, Component diagrams*

Example:

E-commerce store may contain components:

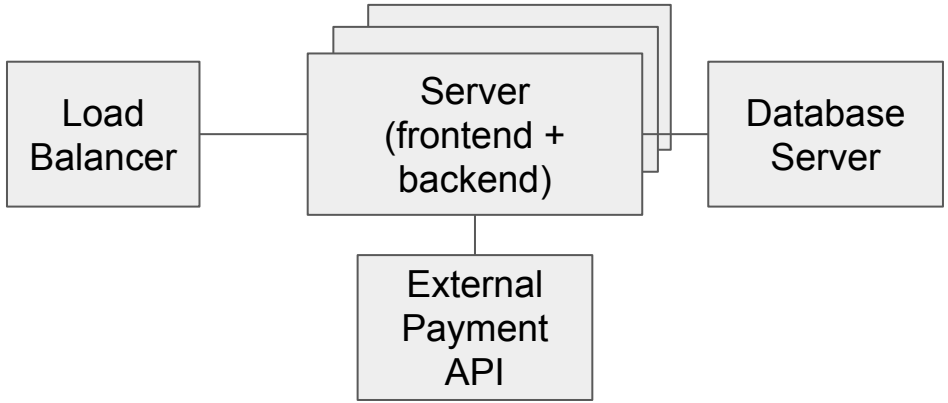
- ProductService: Handles operations related to product search, listing, and management.
  - Java package Product
- CartService: Manages cart-related operations
  - Java package ShoppingCart
- etc.

4+1 architectural view model: Physical view

- Describes the system's physical architecture, including hardware components, network topology, and distribution of software components across different machines or nodes
- Addresses concerns related to deployment, scalability, and performance optimization
- *UML: Deployment diagram*

Example:

- Load balancer distributes traffic between multiple servers
- Servers handle both frontend and backend processes, interact with the database Server and external payment API



4+1 architectural view model

5. Scenarios

- “+1” aspect of the model
- Illustrate how the system functions in real-world situations, using a small set of use cases (scenarios)

(Modern) principles of software architecture

- Separation of Concerns (SoC)
  - Keeping different aspects of the system's functionality or behavior separate and well-defined
  - Each part of the codebase has a single responsibility that makes the code more maintainable and understandable
- Modularity
  - Organizing software into discrete, interchangeable components or modules

Component - based architecture
=> reusability, interoperability, encapsulation, maintainability, scalability, …

- Avoid Big Design Up Front (BDUF)
- Build to change instead of build to last
- Use consistent principles within the components / layers / subsystems
- …

Architectural styles and patterns

Similar to design patterns:

- Provide abstract framework for a family of systems
- Help communication

Architectural styles and patterns

Architecture addresses a wide variety of issues

We have various types of styles/patterns and some of them can be mutually combined

- Deployment
- Structure
- Communication
- Domain
- Network
- ….

Common architectural styles / patterns

Communication models

- Client-server model
- Peer-to-peer model

Service-oriented patterns

- Service-oriented architecture
- Service-oriented architecture with Enterprise Service Bus
- Microservices architecture

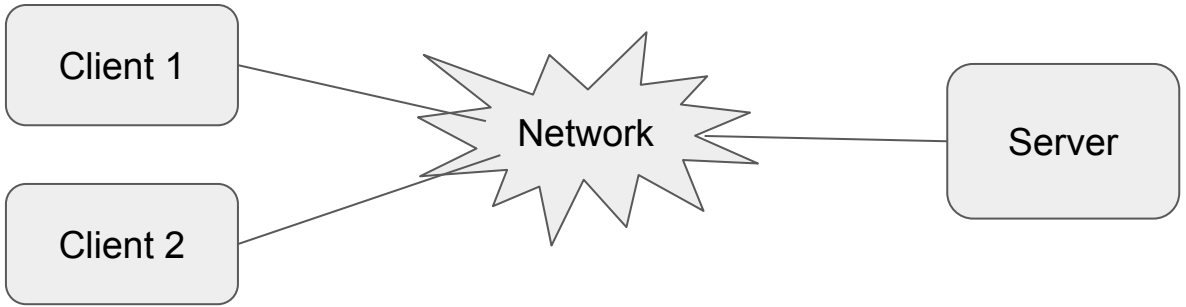
Other

- Layered architecture
- Domain-driven development
- Model-view-controller

Communication models: Client-server model

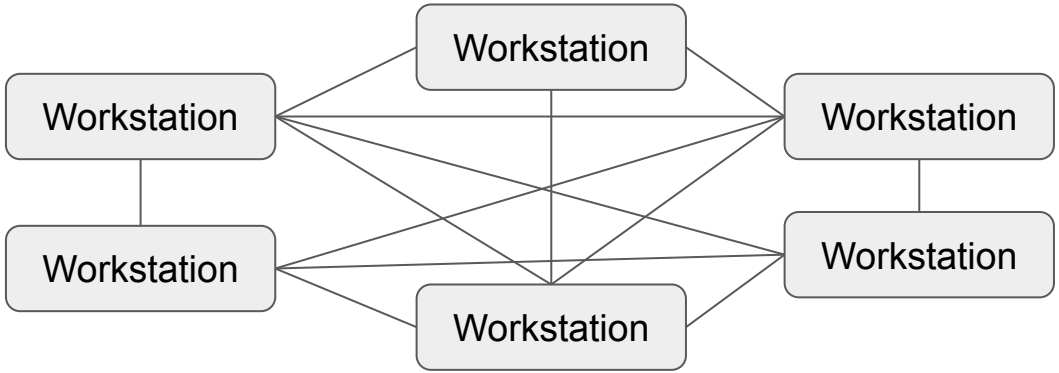
- Used in networking / distributed systems
- One or more clients connected to a server over a network or internet connection.
- The server hosts, delivers and manages most of the resources and services to be consumed by the client
- Typically follows a request-response pattern.

- Example:
  web browsing

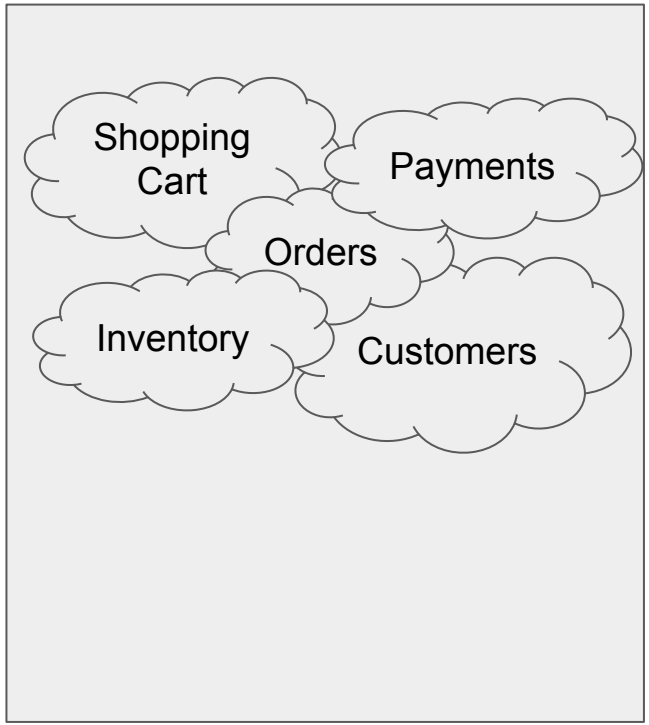


Communication models: Peer-to-peer (P2P) model

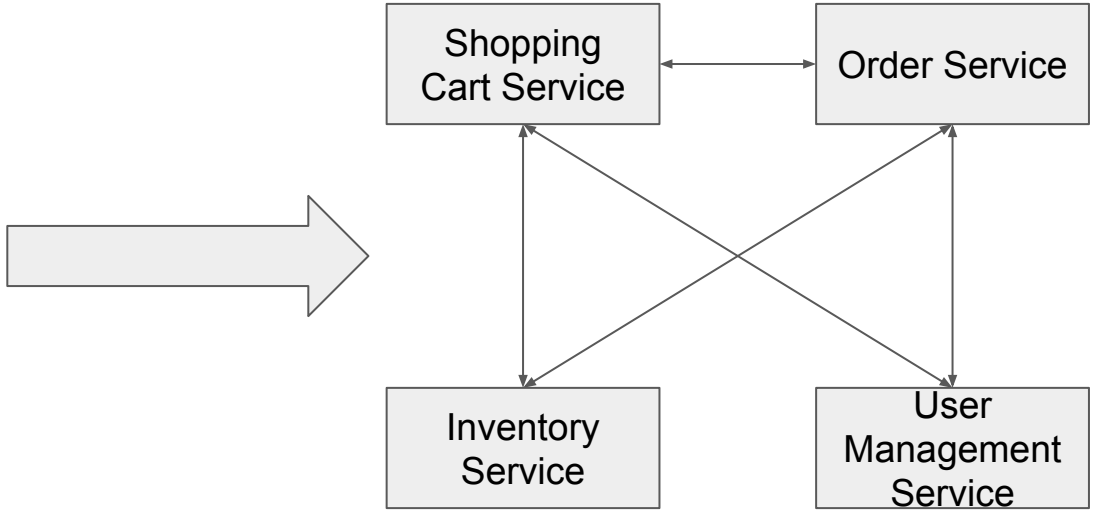
- Used for distributed systems
- Computers, devices, or nodes within a network (peers) communicate and **collaborate directly** with each other without the need for a centralized server or hierarchy of control
- Each peer has equivalent capabilities, and they can act both as clients and servers, sharing resources, data, or services with one another.



Service-oriented architecture (1)



Monolith



Service-oriented architecture (SOA)

Service-oriented architecture (2)

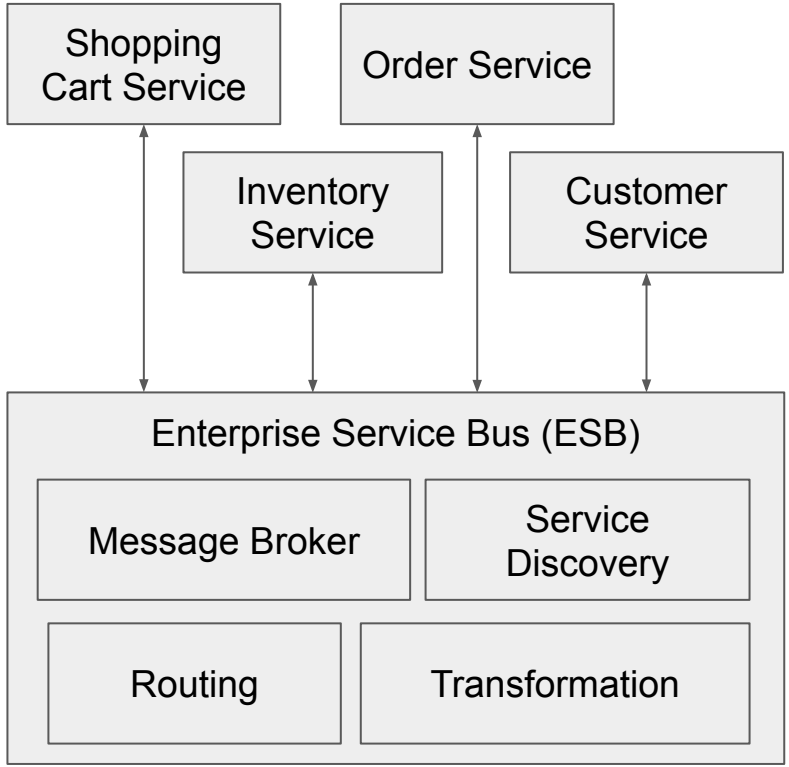
- A specialization of component-based architecture where **software components** are **services**
- *A service*
  1. Is a logical representation of a repeatable business activity that has a specified outcome
  2. Is self-contained
  3. May be composed of other services
  4. Is a “black box” to consumers of the service

- Properties of services: business-oriented, interface-based, discoverable and invokable, distributed, loosely-coupled

Service-oriented architecture with ESB

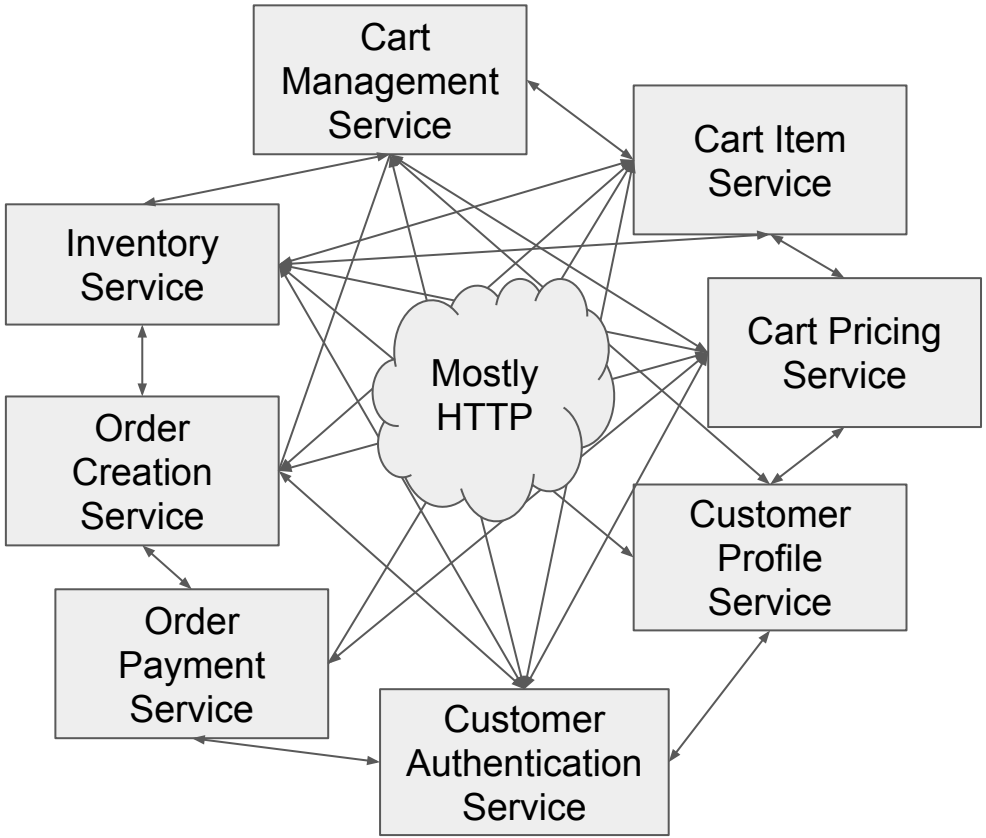
- A variant of SOA
- Enterprise Service Bus = central integration point
  - Routing messages between services, monitoring and control, security, …

SOA generally may not contain ESB !



Microservice architecture

- A variant of SOA
- Loosely coupled, **fine-grained** services
- Microservices focus on one thing and operate **independently**
- Mutual communication over well-defined APIs
- Commonly used in cloud-native applications



Domain Driven design (DDD)

- The software systems is build “around” the core domain knowledge and concepts of a business
- The structure and language of software code (class names, class methods, class variables) and data entities **should match the business domain**
- Example: if software processes loan applications, it might have classes like *"loan application"*, *"customers"*, and methods such as *"accept offer"* and *"withdraw*".

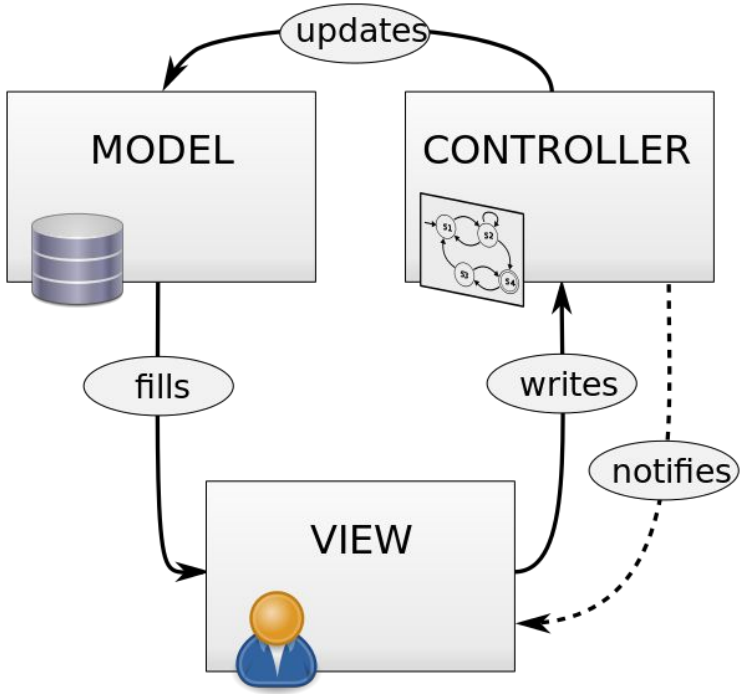
Layered architecture

- Components within the layered architecture pattern are organized into **horizontal layers**
- Each layer performs a specific role within the application (-> SoC)
- 3-layer architecture (logical separation, but not necessarily physical):
  - Presentation layer (UI layer)
  - Application (business) layer
  - Data access layer
- Physical separation - 3-tier architecture - example:
  - Presentation layer (UI layer) - web browser
  - Application layer - web server(s)
  - Data access layer - database server(s)
- Frontend, backend

Model-view-controller

= pattern used commonly for developing user interfaces

- **The model** manages the data of the application.
- **The view** renders presentation of the model in a particular format (chart, table, ..)
- **The controller** processes the user input and updates the data model objects.



Resources

[1] SWEBOOK v3

[2] Ian Sommerville: Software Engineering (10th edition)

[3] Robert Lukotka: Architecture

[4] Martin Fowler: Design Stamina Hypothesis

[5] Wikipedia: Domain-driven design

[6] Philippe Kruchten: Architectural Blueprints - The “4+1” View Model of Software Architecture, 1995.

[7] MVC Diagram (Model-View-Controller) byXinfe, CC BY-SA 3.0

[8] Service Oriented Architecture : What Is SOA?, The Open Group SOA Working Group.

[9] Michal Kostič: Service-oriented architectural patterns

[10] Eric Evans: Domain-Driven Design: Tackling Complexity in the Heart of Software, 2003.