Design

10/17/2023

Design

- It’s where you stand with a foot in two worlds-
    - The world of technology
    - The world of people and human purposes

And

you try to bring the two together

---- Mitch Kapor

Work Product from Design

Work Product from Design

- A design model that encompasses architectural, interface, component-level, and deployment representations

Work Product from Design

- A design model that encompasses architectural, interface, component-level, and deployment representations
- This model can be assessed for quality and improved
    - Code is generated
    - Tests are conducted
    - End users become involved in large numbers

Work Product from Design

- A design model that encompasses architectural, interface, component-level, and deployment representations
- This model can be assessed for quality and improved
  - Code is generated
  - Tests are conducted
  - End users become involved in large numbers

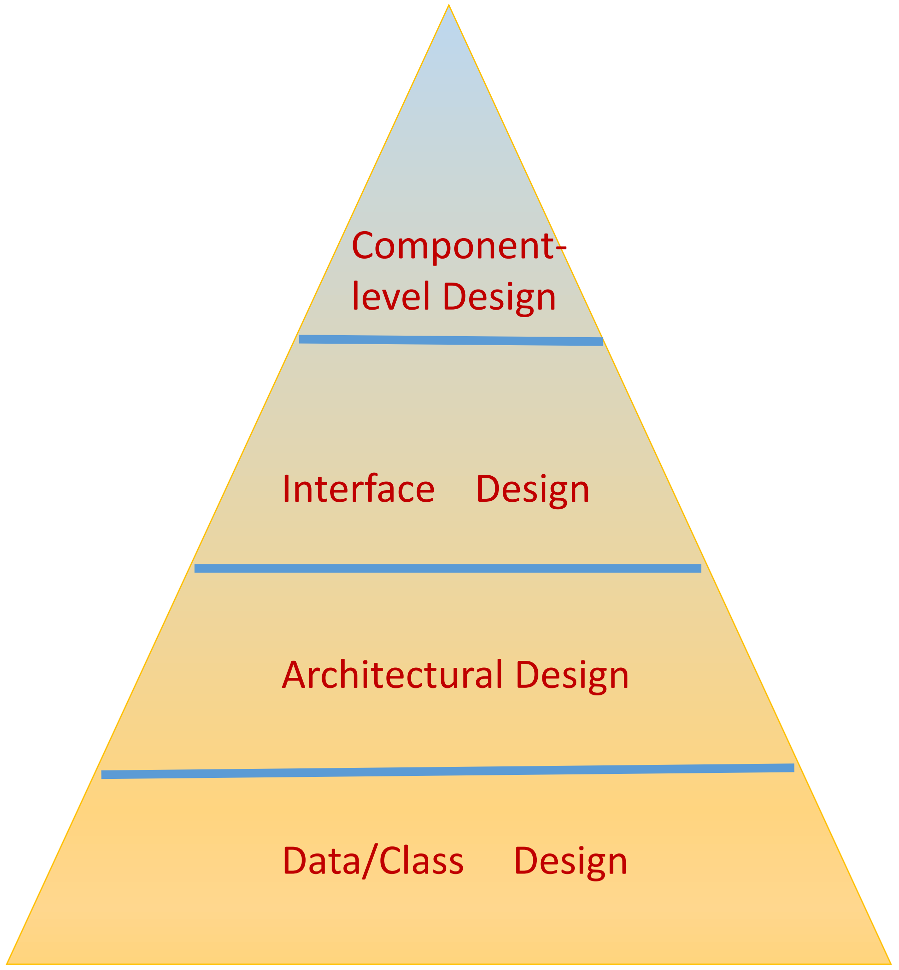
Design is the place where software quality is established.

Steps for Design

- The architecture of the system or product must be presented.
- The interfaces that connect the software to end users, to other systems and device, and to its own constituent components are modeled.
- The software components that are used to construct the stem are designed.

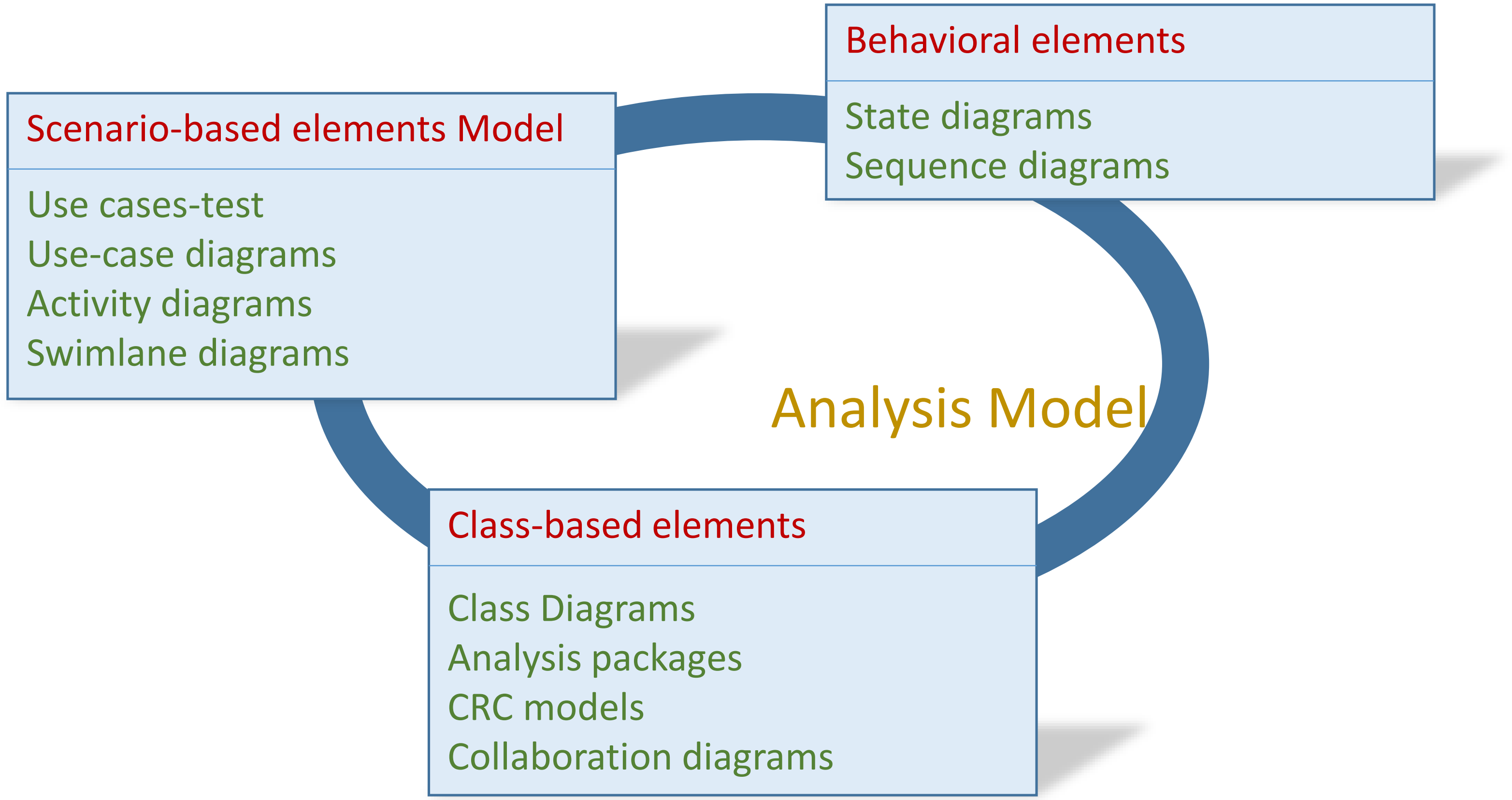
Steps for Design

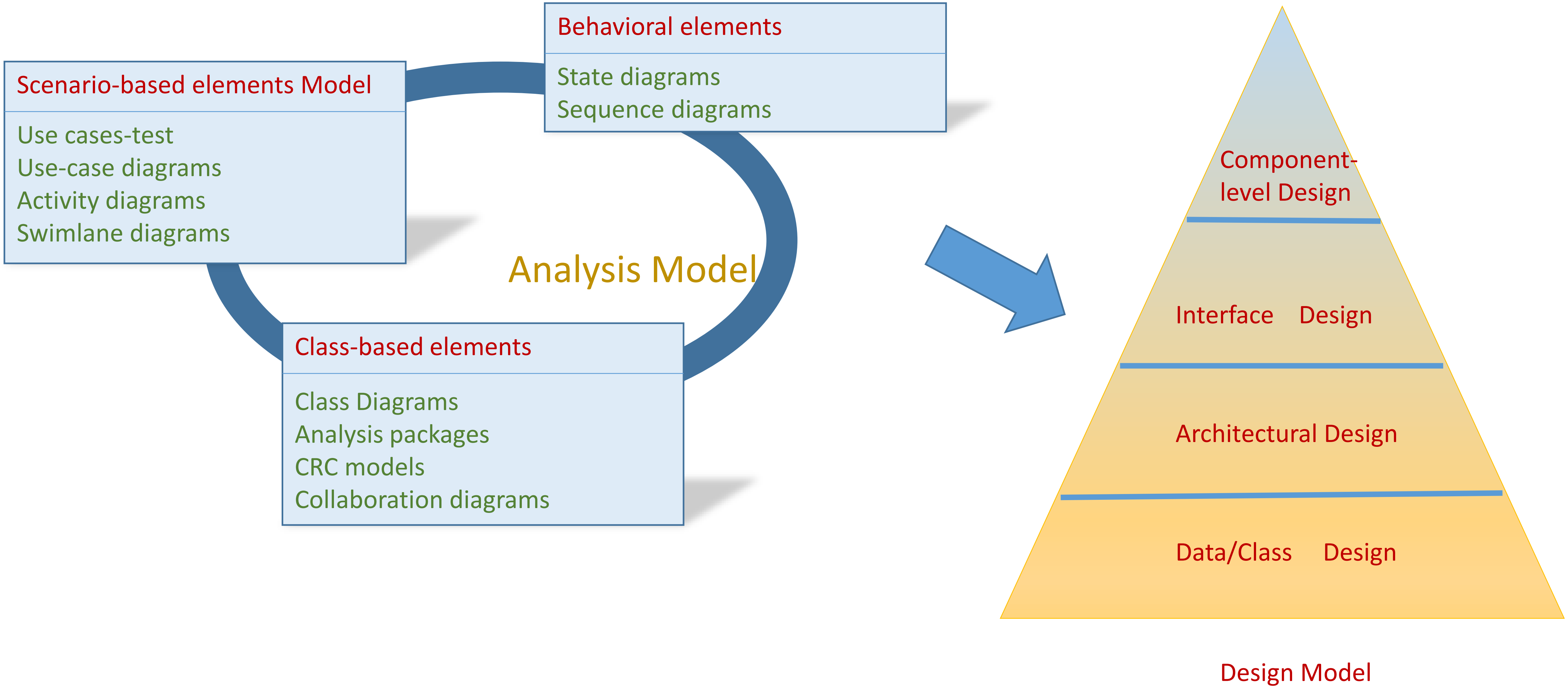
- The architecture of the system or product must be presented.
- The interfaces that connect the software to end users, to other systems and device, and to its own constituent components are modeled.
- The software components that are used to construct the stem are designed.



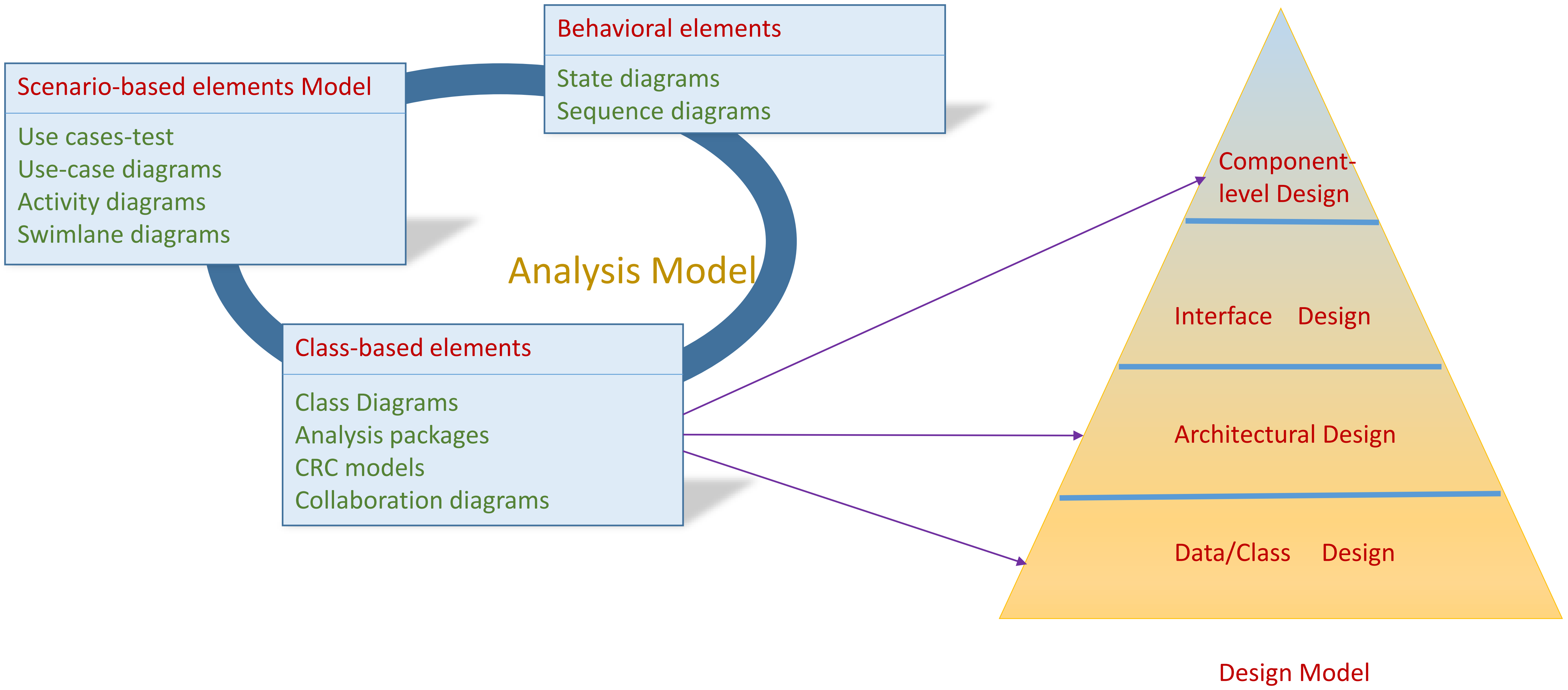
Design Model

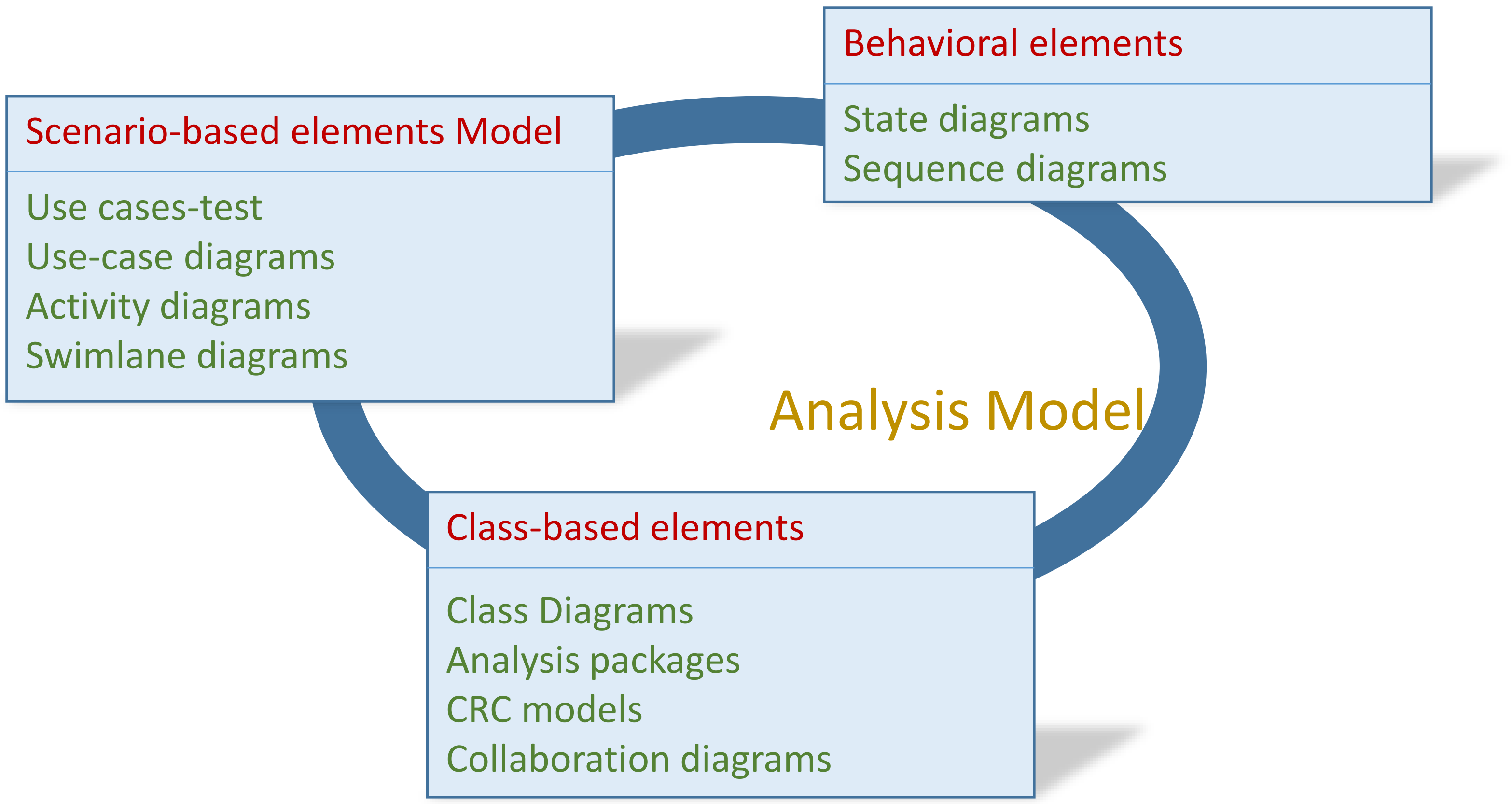
Recall Analysis Model

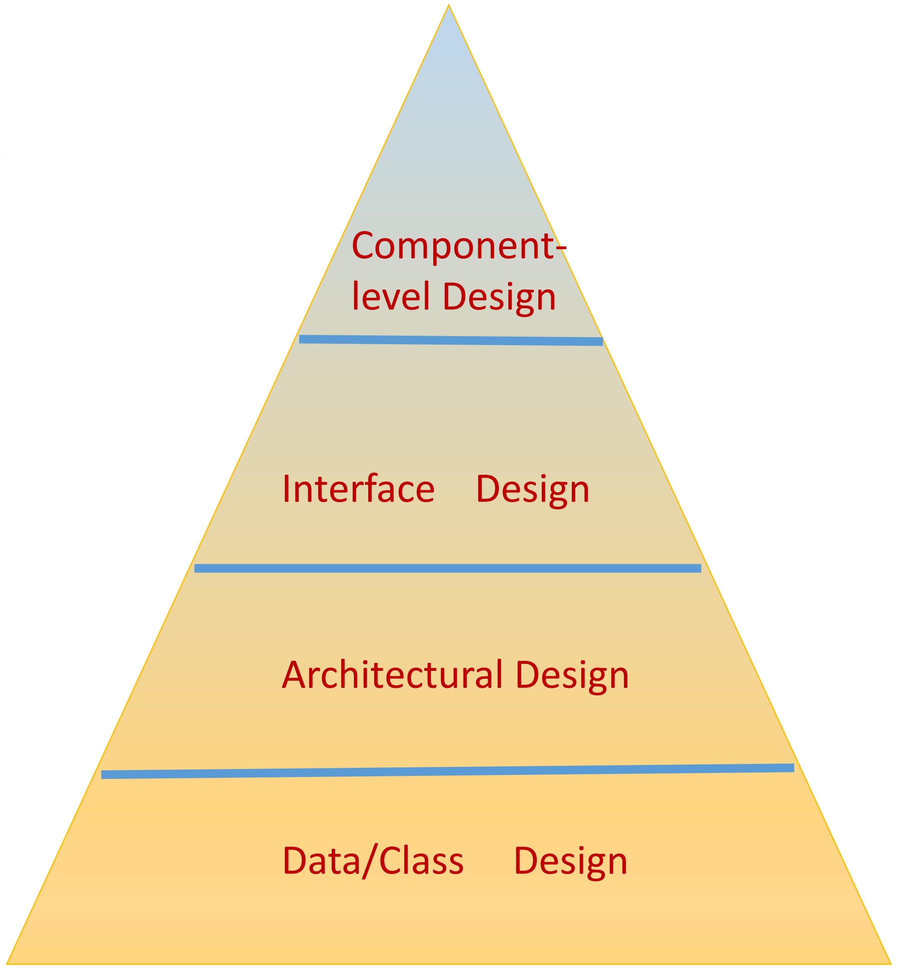




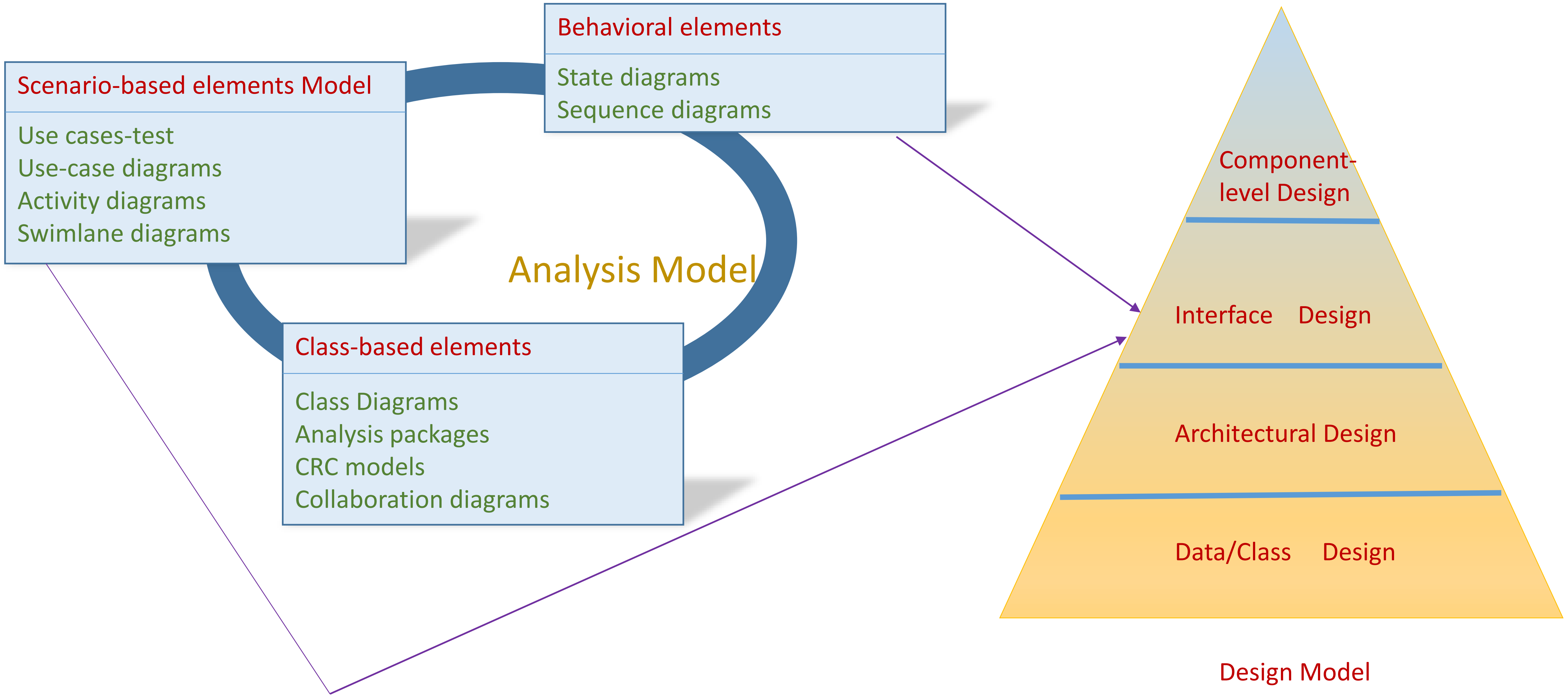
Scenario-based elements Model
Use cases-test
Use-case diagrams
Activity diagrams
Swimlane diagrams
Behavioral elements
State diagrams
Sequence diagrams
Analysis Model
Class-based elements
Class Diagrams
Analysis packages
CRC models
Collaboration diagrams
Component-level Design
Interface Design
Architectural Design
Data/Class Design
Design Model

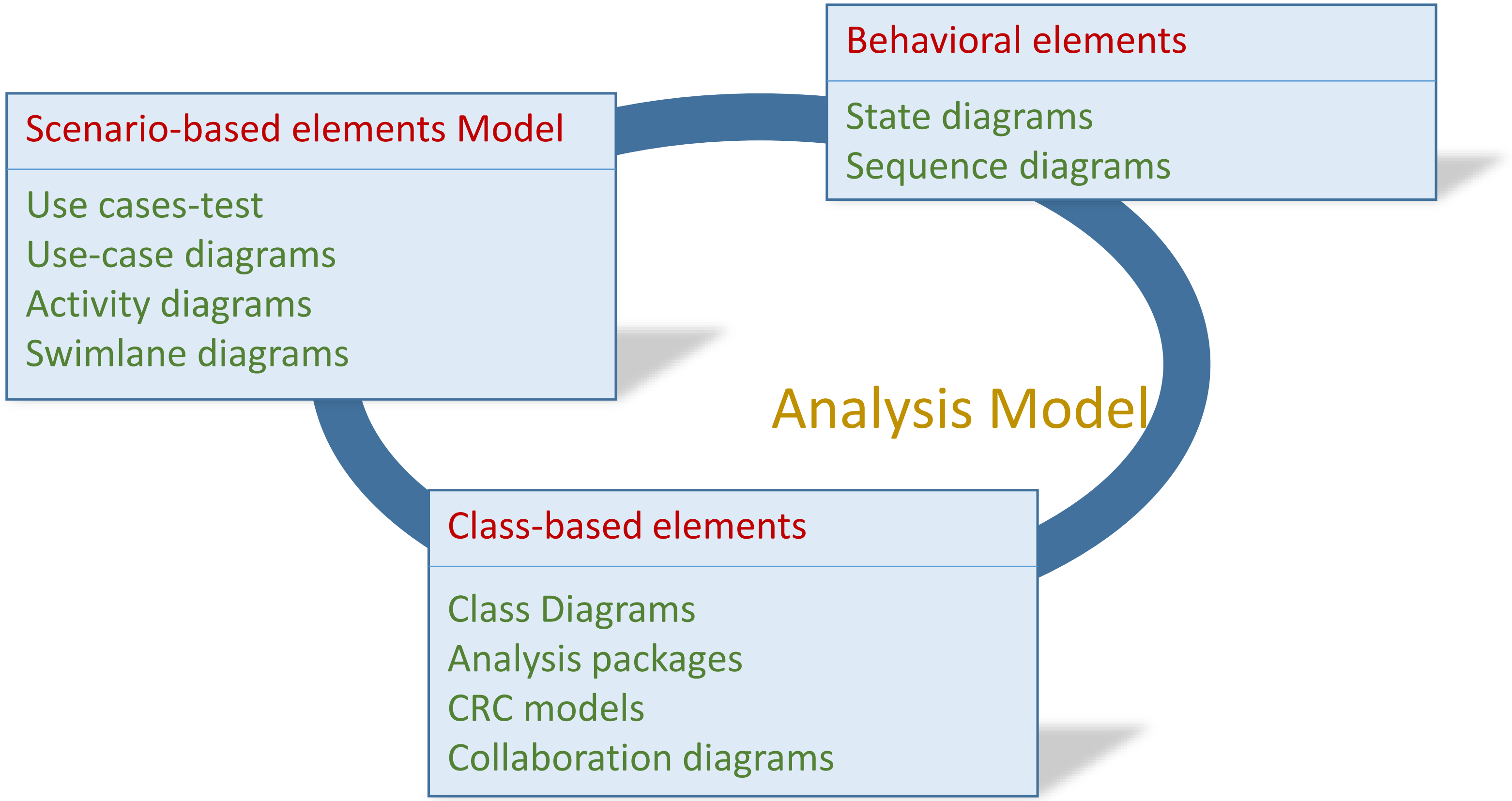

Behavioral elements
State diagrams
Sequence diagrams
Scenario-based elements Model
Use cases-test
Use-case diagrams
Activity diagrams
Swimlane diagrams
Analysis Model
Class-based elements
Class Diagrams
Analysis packages
CRC models
Collaboration diagrams
Component-level Design
Interface Design
Architectural Design
Data/Class Design
Design Model

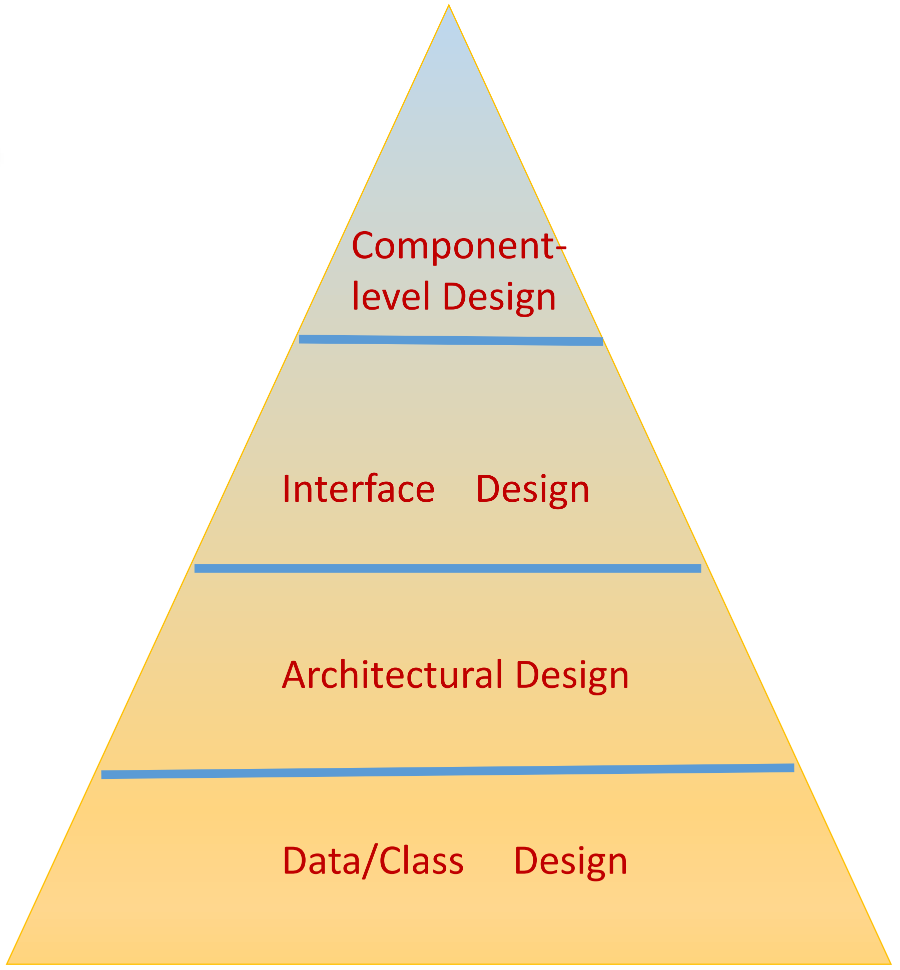




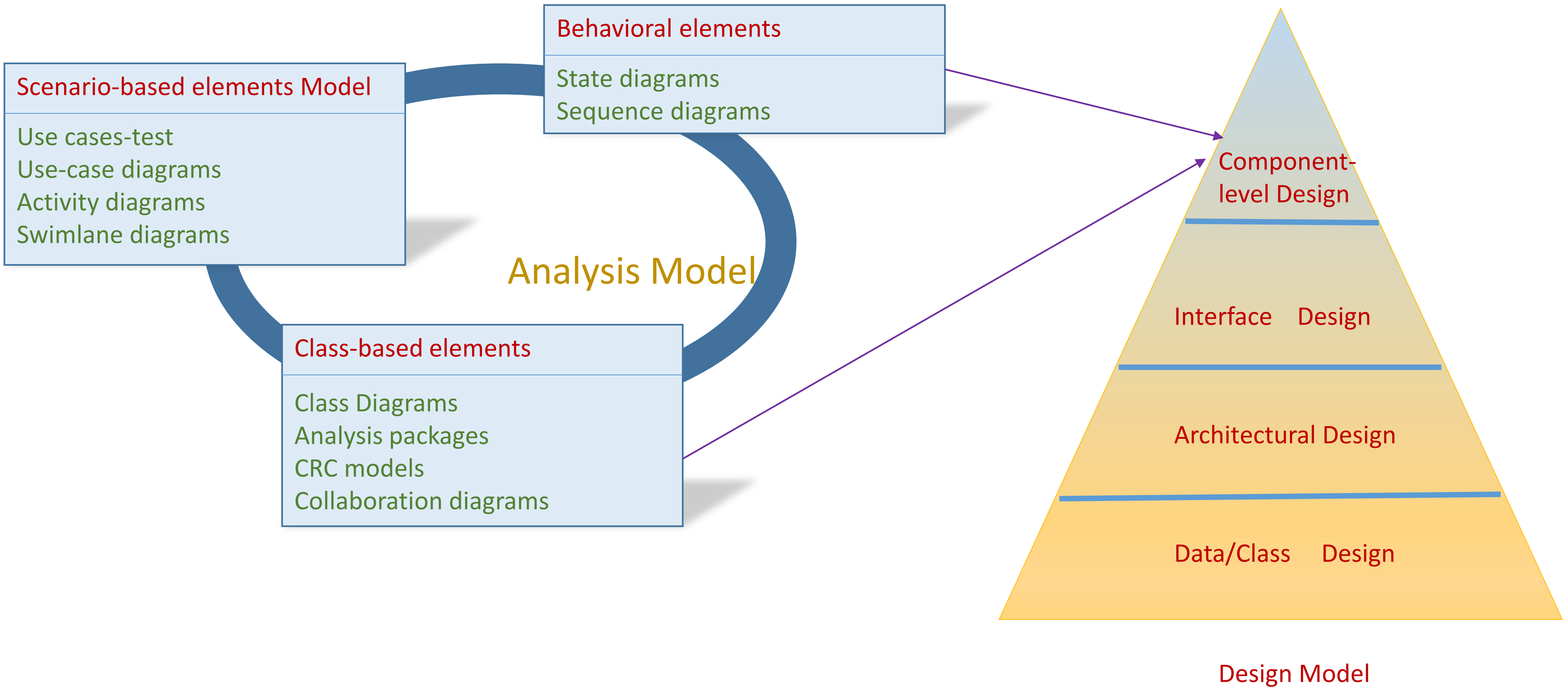
Design Model


Scenario-based elements Model
Use cases-test
Use-case diagrams
Activity diagrams
Swimlane diagrams
Behavioral elements
State diagrams
Sequence diagrams
Class-based elements
Class Diagrams
Analysis packages
CRC models
Collaboration diagrams
Analysis Model
Component-level Design
Interface Design
Architectural Design
Data/Class Design
Design Model





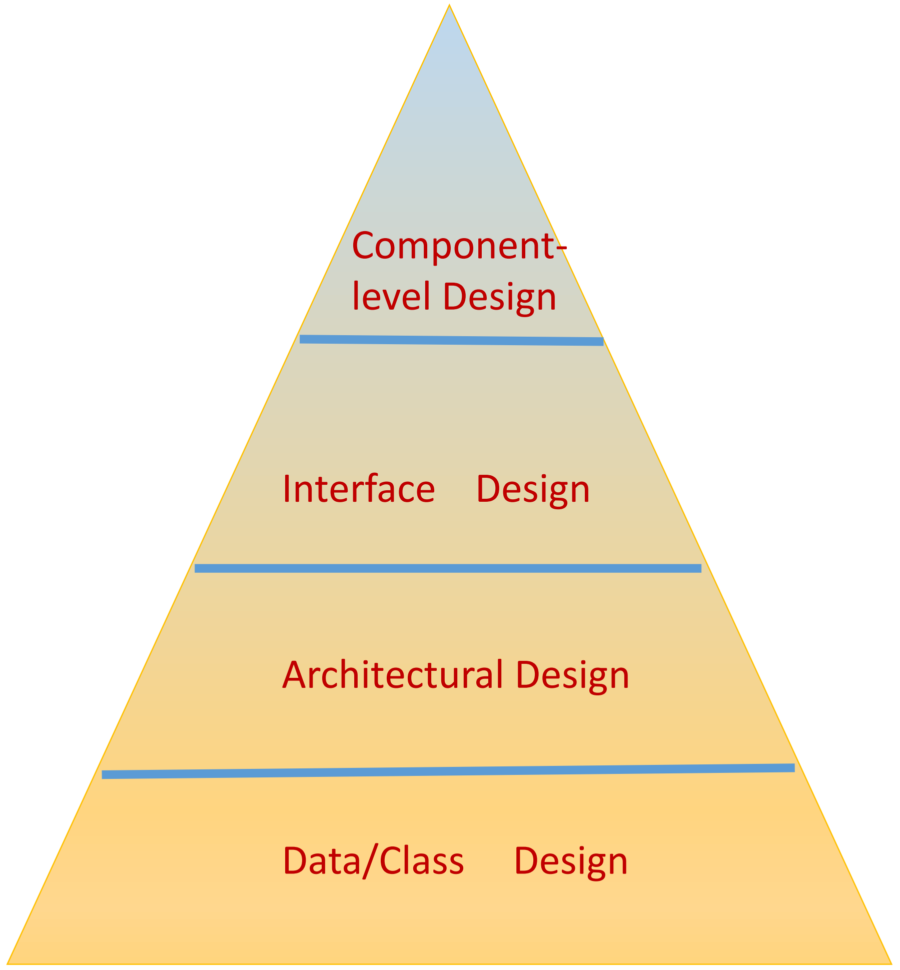
Design Model


Scenario-based elements Model
Use cases-test
Use-case diagrams
Activity diagrams
Swimlane diagrams
Behavioral elements
State diagrams
Sequence diagrams
Analysis Model
Class-based elements
Class Diagrams
Analysis packages
CRC models
Collaboration diagrams
Component-level Design
Interface Design
Architectural Design
Data/Class Design
Design Model

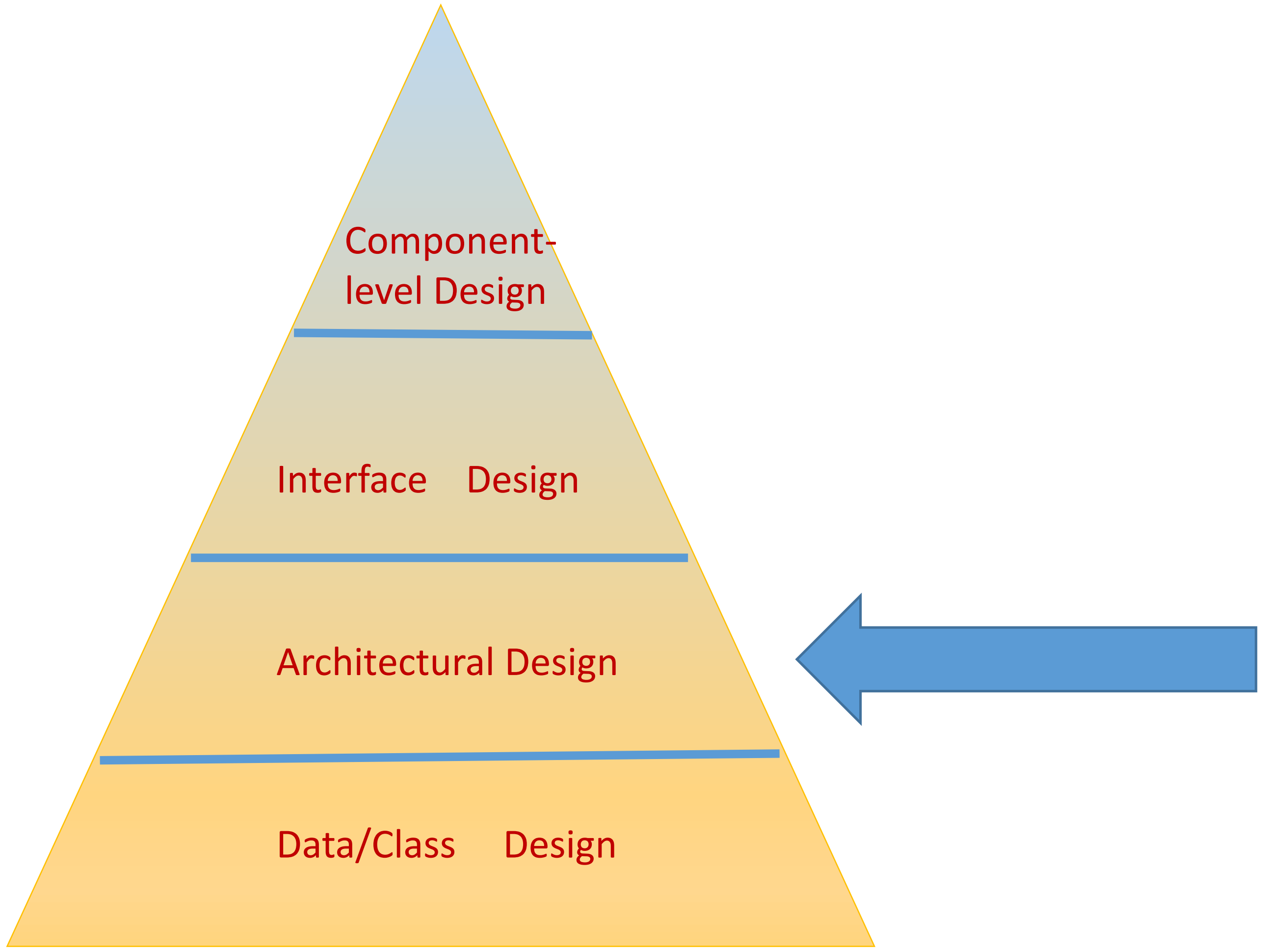
Work Product from Design

- A design model that encompasses architectural, interface, component-level, and deployment representations
- This model can be assessed for quality and improved
  - Code is generated
  - Tests are conducted
  - End users become involved in large numbers

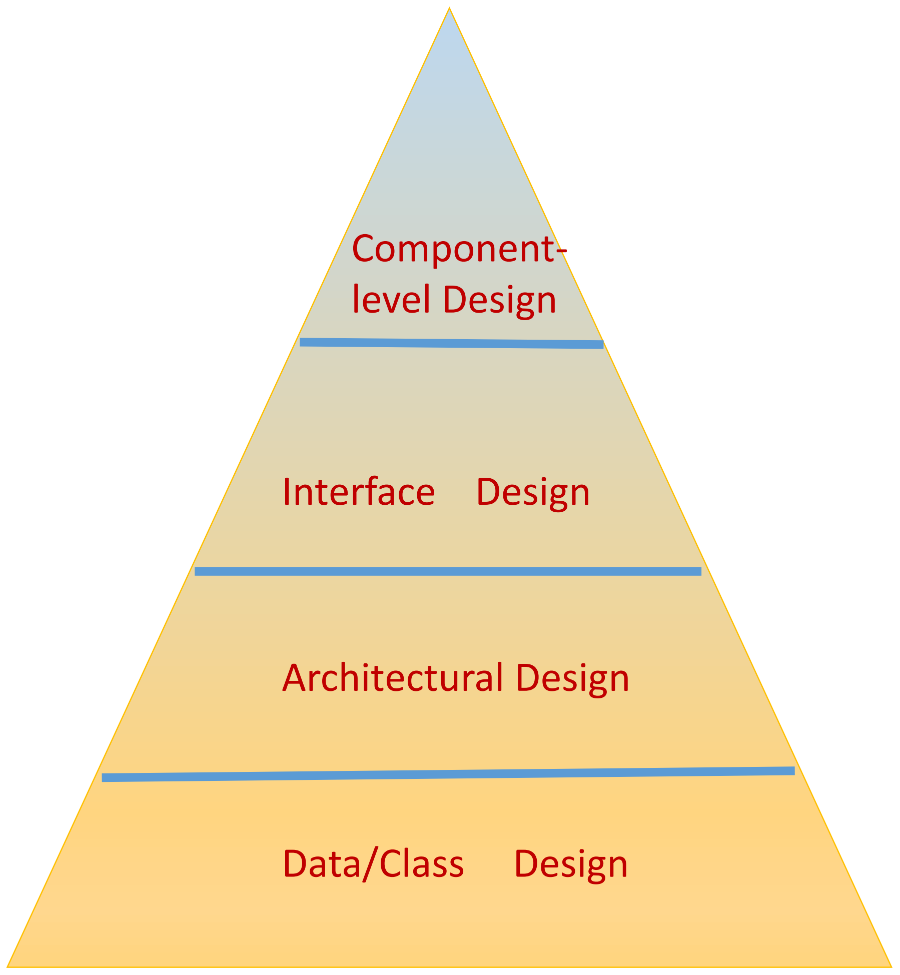
Design is the place where software quality is established.



Design Model

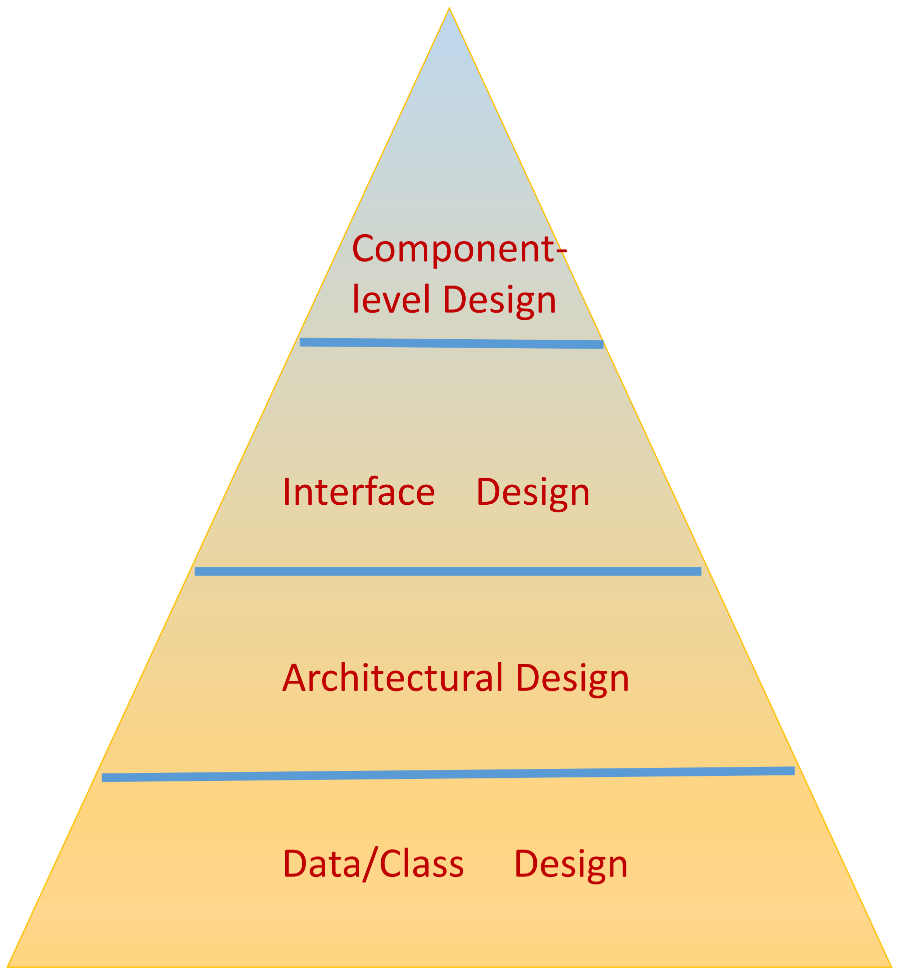


Design Model



Design Model

- What is software architecture?
- What is the output of architectural design?



Design Model

- What is software architecture?
  - Organization, structure
- What is the output of architectural design?
  - Components
  - The interaction among components

Why should we do architectural design

- The big picture before you worry about details
    - Functional requirements

Why should we do architectural design

- The big picture before you worry about details
  - Functional requirements

- Non-functional requirements

Why should we do architectural design

- The big picture before you worry about details
    - Functional requirements

- Non-functional requirements
    - Maintainability
        - The less interaction among components, the better
    - Availability
        - Redundant component can be used to improve availability
    - Performance

How to design

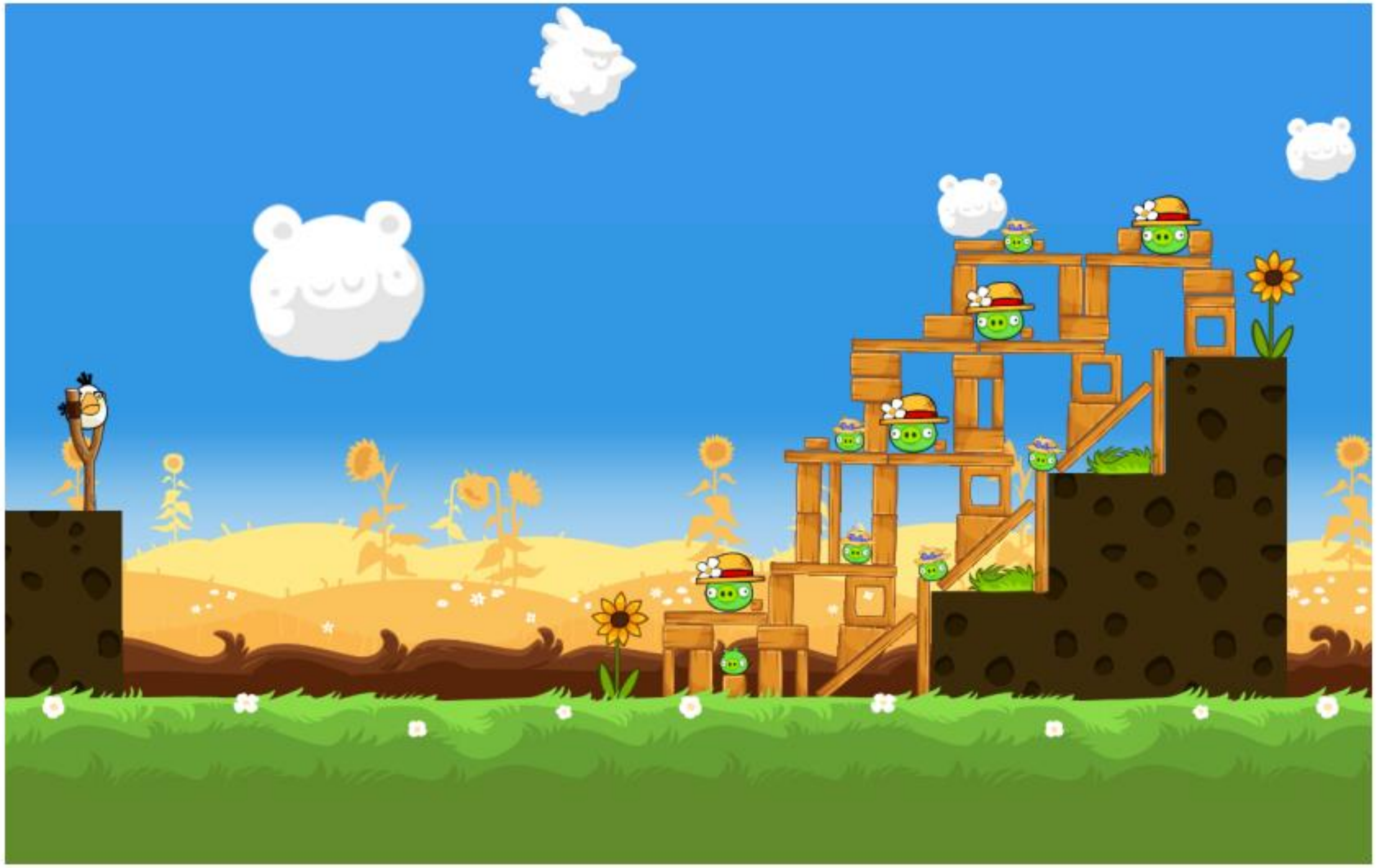
- Follow the requirement and modeling
  - Architectural design can be conducted right after use-case study

How to design

- Follow the requirement and modeling
  - Architectural design can be conducted right after use-case study

- Architecture styles/patterns

Angry Birds!

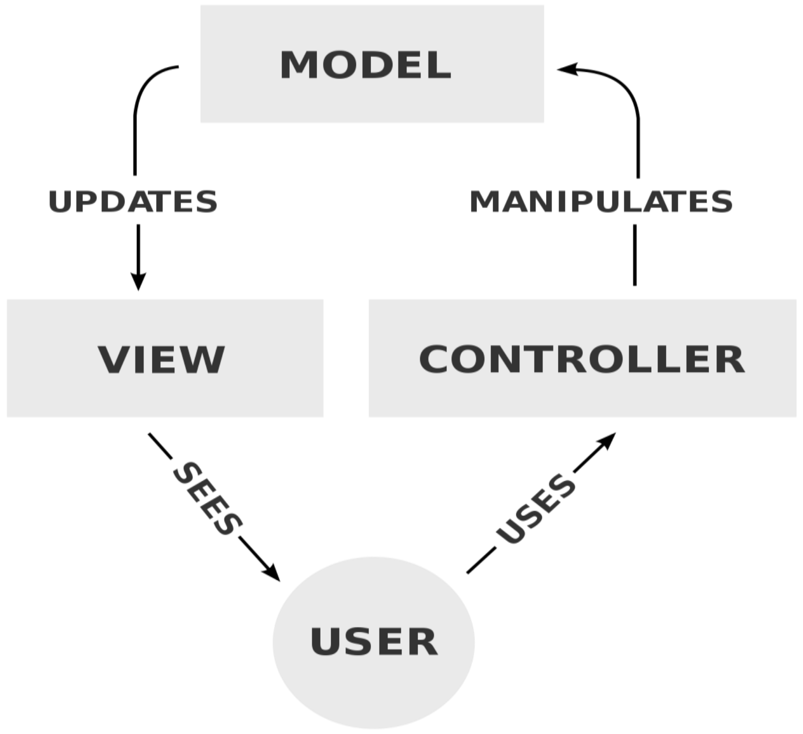


Model-View-Controller patterns

- It separates the application logic from the user interface and the control between the user interface and application logic.

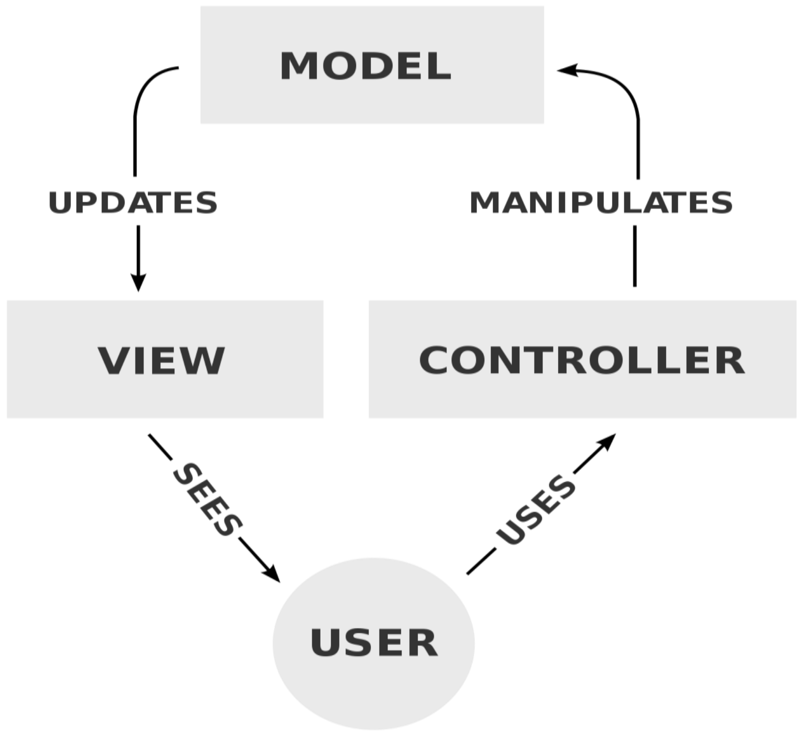
Model-View-Controller patterns

- It separates the application logic from the user interface and the control between the user interface and application logic.



Model-View-Controller patterns

- It separates the application logic from the user interface and the control between the user interface and application logic.

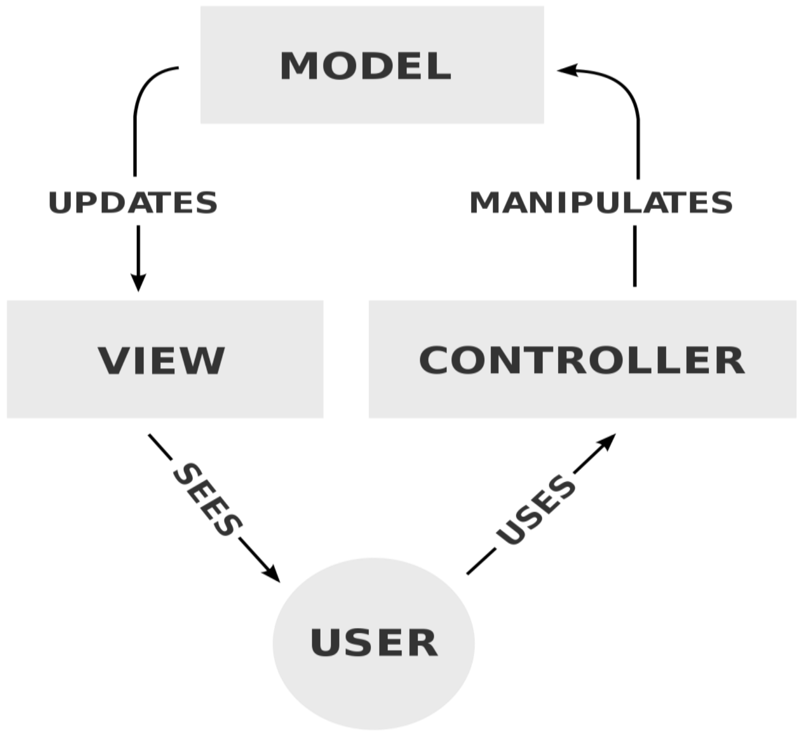


All about separation of concerns.

- Low coupling between the Model and the View/controller layers

Model-View-Controller patterns

- It separates the application logic from the user interface and the control between the user interface and application logic.

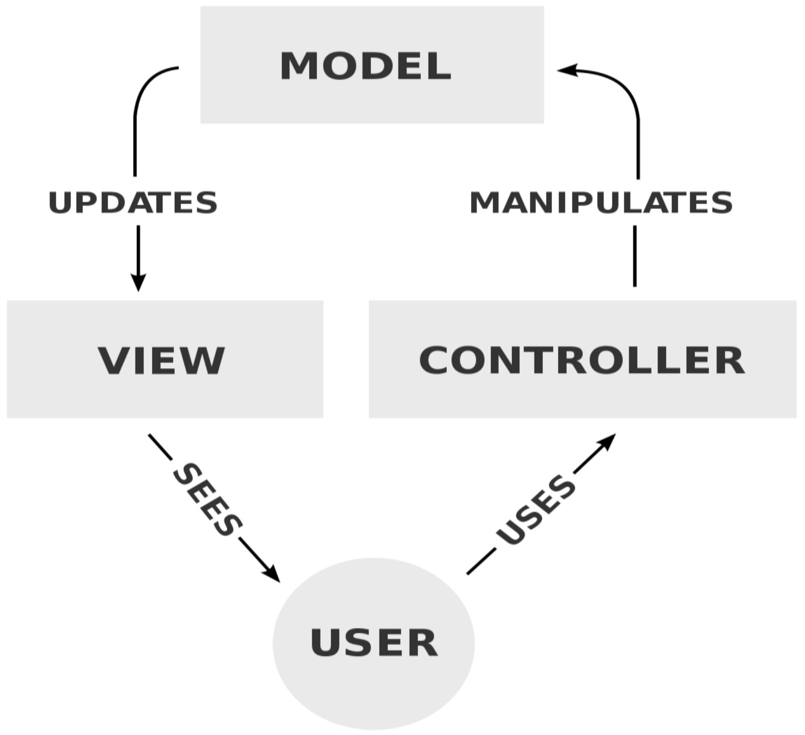


All about separation of concerns.

- Low coupling between the Model and the View/controller layers
- The direction in which those connection goes:
  - All instructions flow from the view/control to the model
  - The model NEVER tells the view/controller what to do

Model-View-Controller patterns

- It separates the application logic from the user interface and the control between the user interface and application logic.

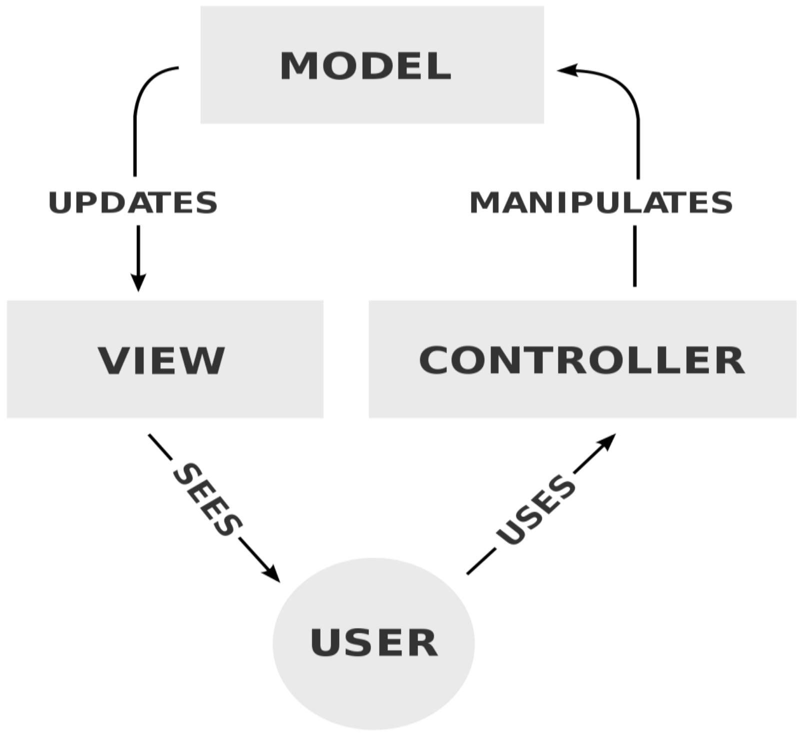


All about separation of concerns.

- Low coupling between the Model and the View/controller layers
- The direction in which those connection goes:

Model-View-Controller patterns

- It separates the application logic from the user interface and the control between the user interface and application logic.



All about separation of concerns.

- Low coupling between the Model and the View/controller layers
- The direction in which those connection goes:
- The view/controller is permitted to know a little about he Model(specifically, the Model's API), but the Model is not allowed to know anything about the view/controller

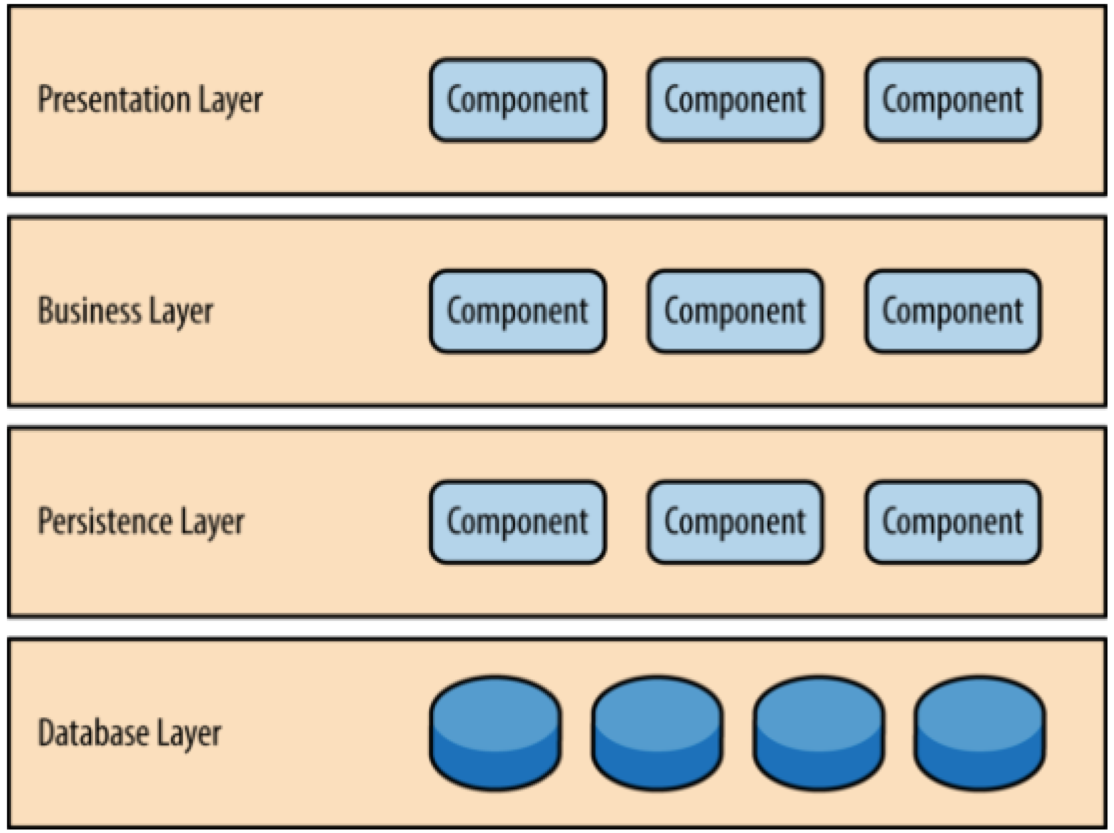
Design concerns in MVC

- Where to put M,V,C, given multiple nodes?

Design concerns in MVC

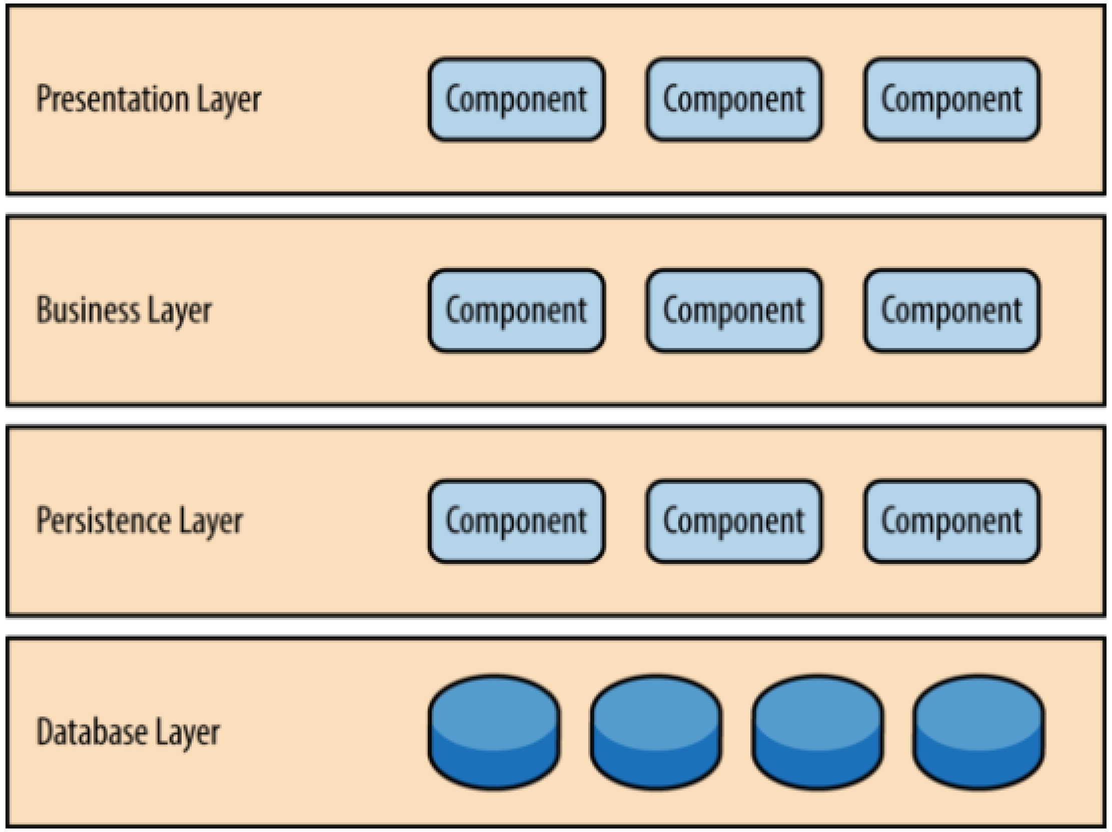
- Where to put M,V,C, given multiple nodes?
  - M is most suitable for server machines
  - C is most suitable for client machines
  - V depends on network bandwidth between server and client

Layered pattern



Layered pattern

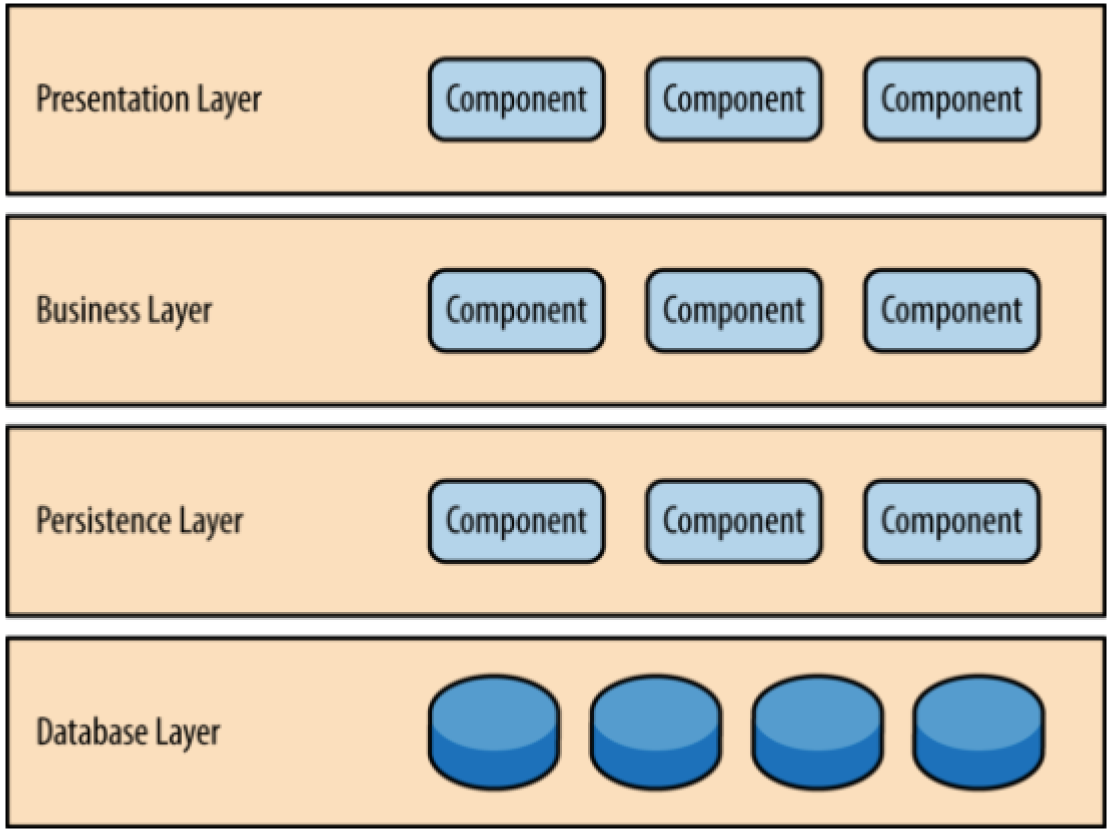
de facto standard for most Java EE applications



Separation of concerns among components.

Layered pattern

de facto standard for most Java EE applications

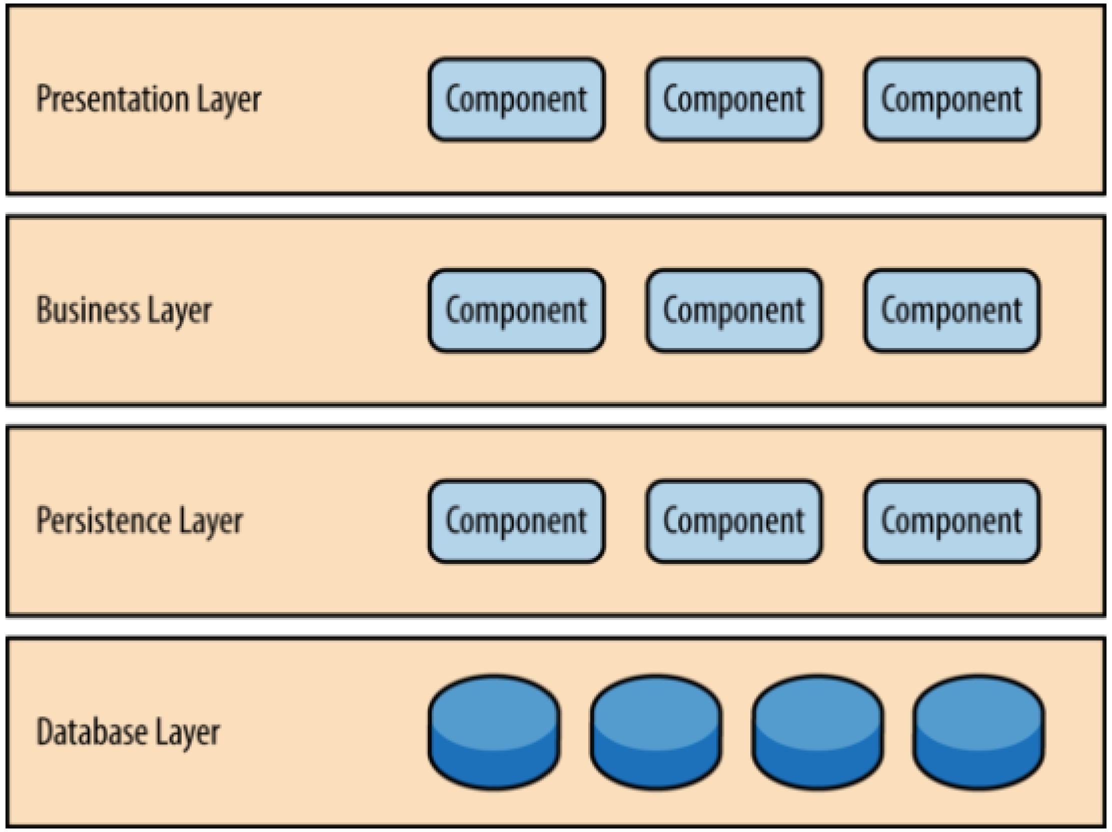


Separation of concerns among components.

- Well-defined component interfaces
- Limited component scope
- Make it easy to build effective roles and responsibility model
- Make it easy to develop, test, govern and maintain applications
    - Modify one layer won't affect the whole system

Layered pattern

de facto standard for most Java EE applications

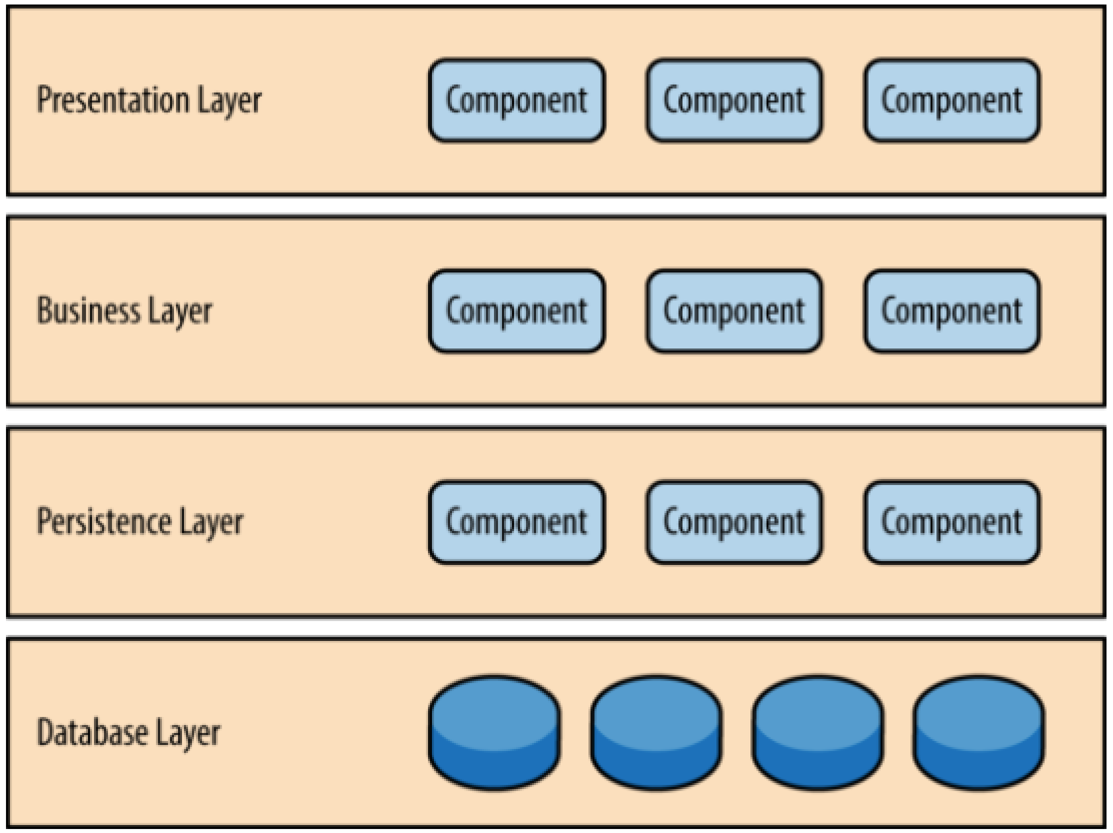


Separation of concerns among components.

Weakness:

- Strict layering may be difficult in practice
- Performance

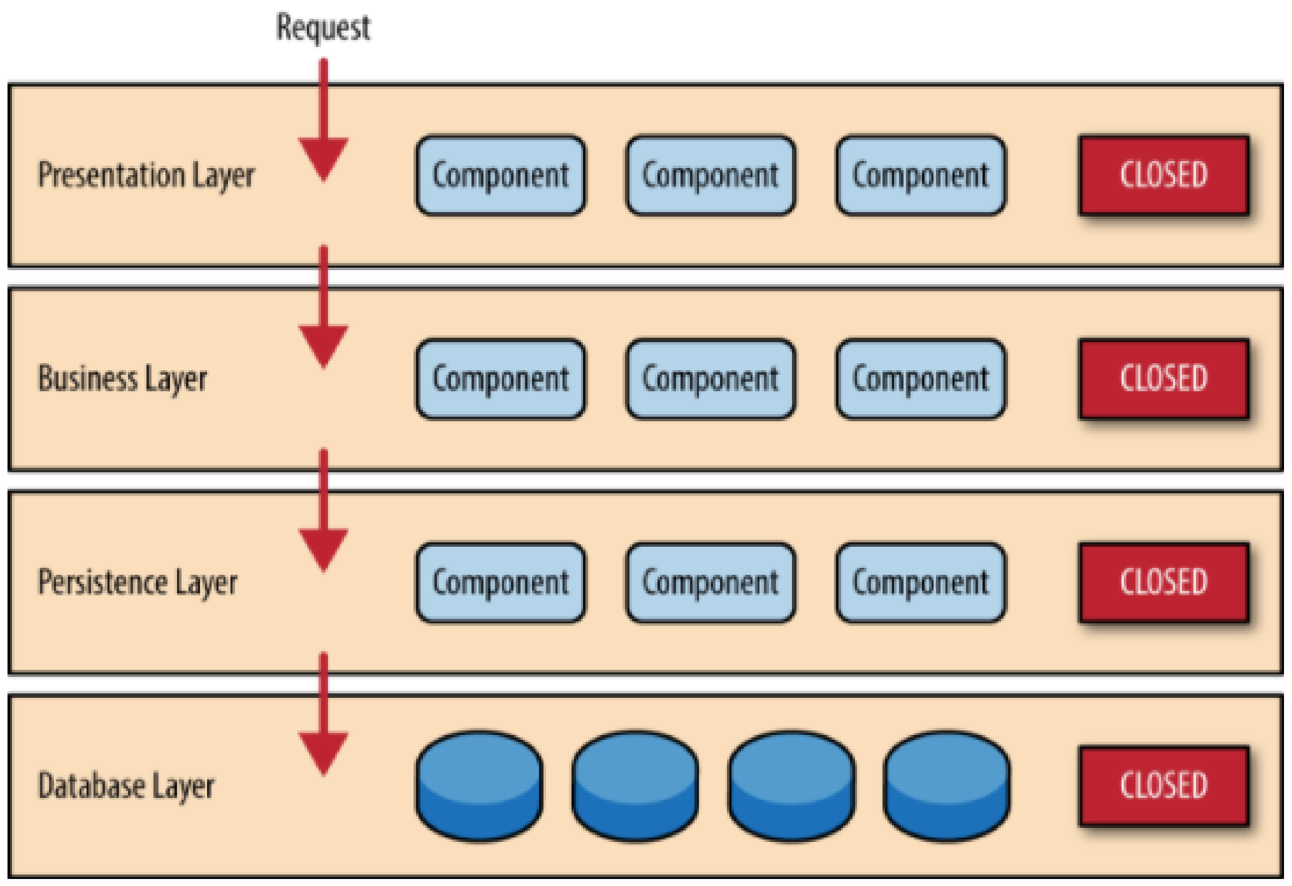
Layered pattern



Separation of concerns among components.

- closed

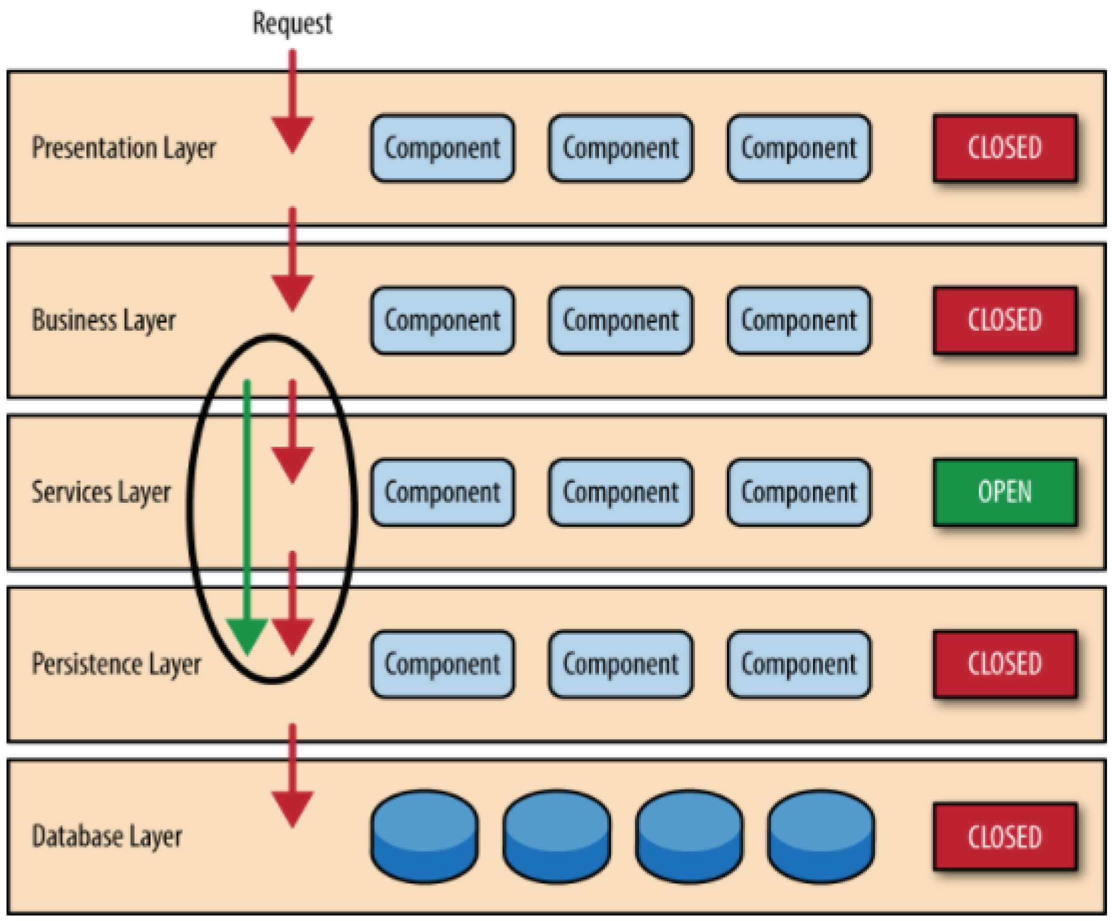
Layered pattern



Separation of concerns among components.

- Closed
  - As a request moves from layer to layer, it must go through the layer right below it to get to the next layer below that one.

Layered pattern



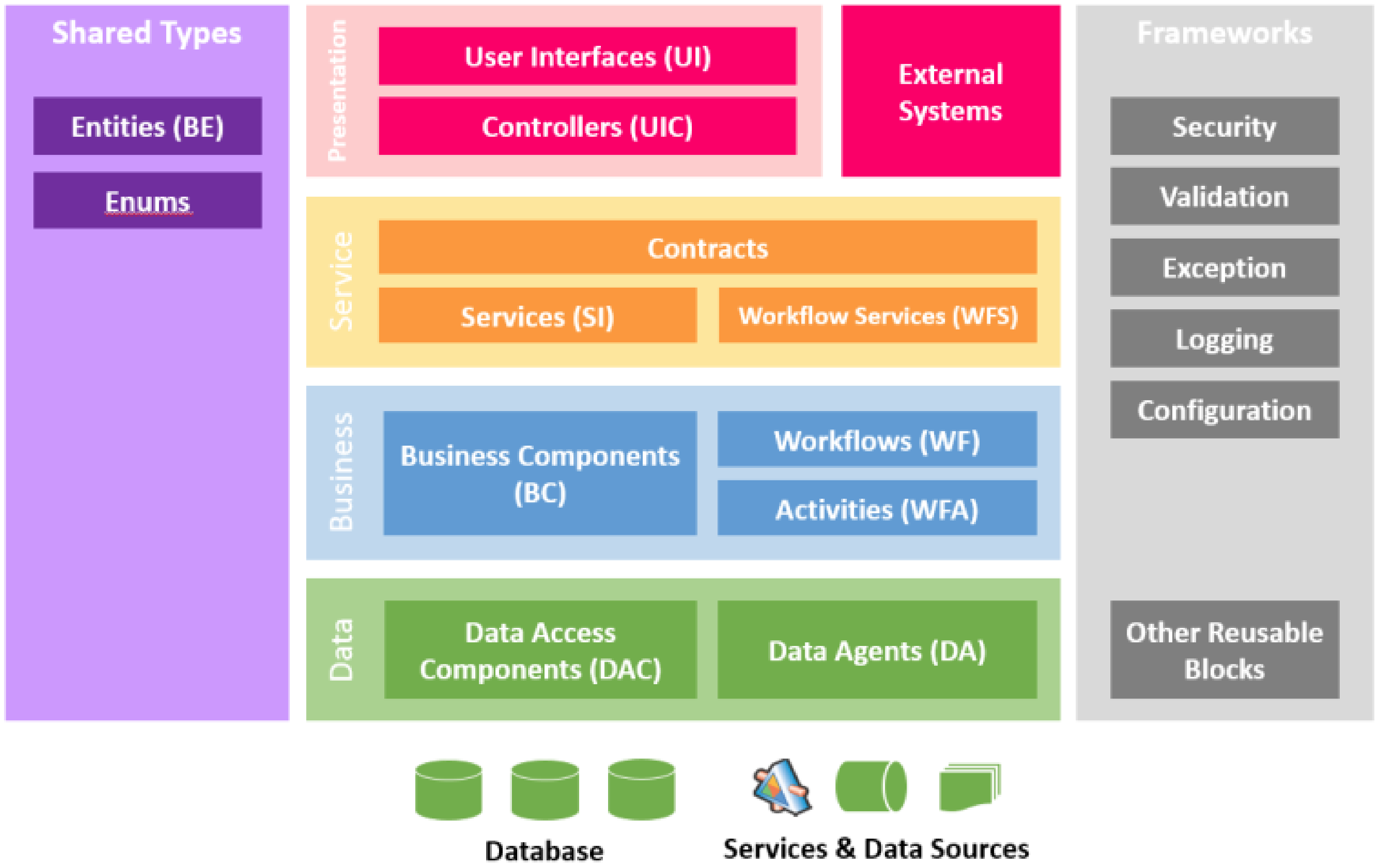
Separation of concerns among components.

- Closed
    - As a request moves from layer to layer, it must go through the layer right below it to get to the next layer below that one.
- Open

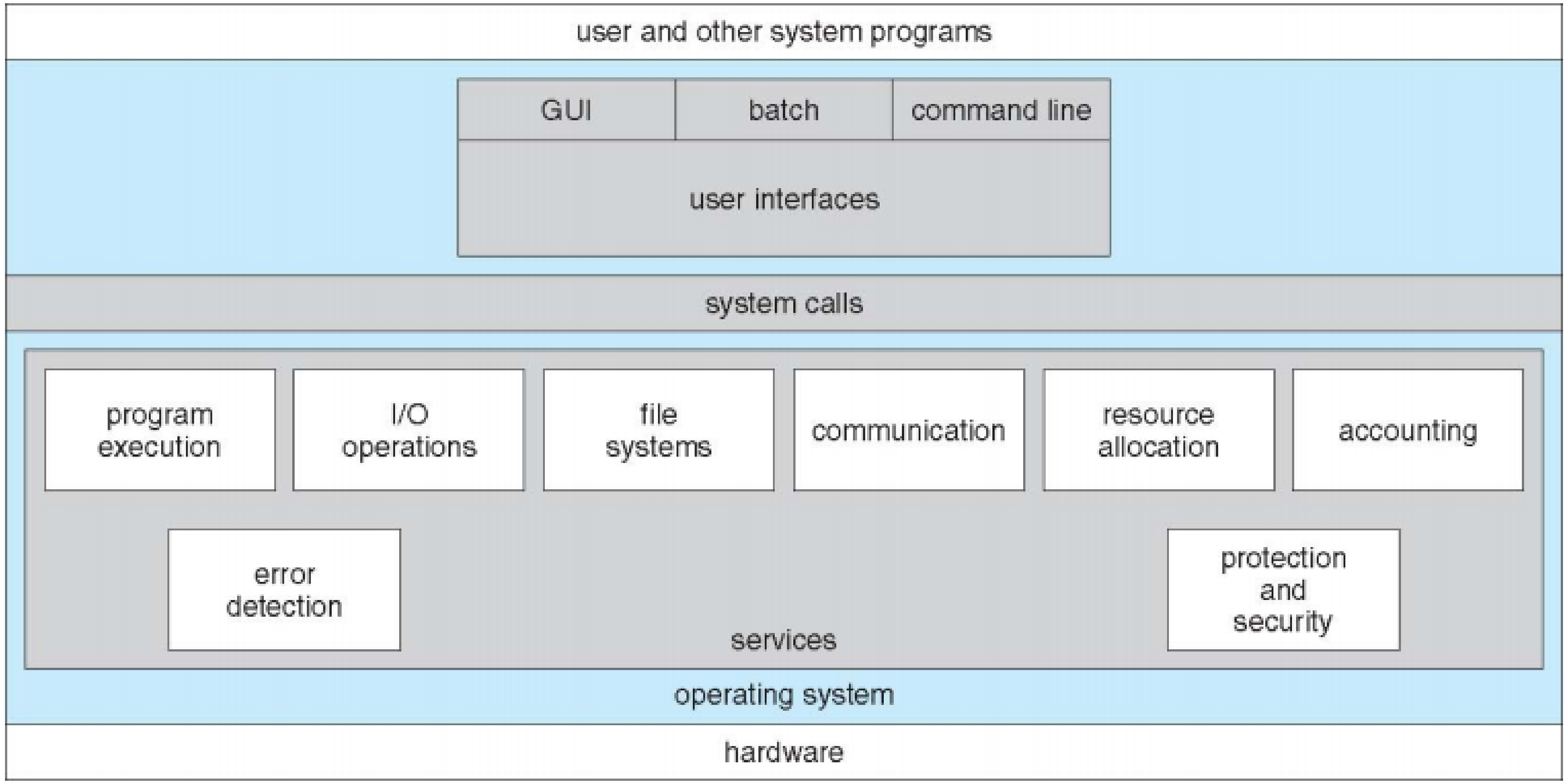
Layered Design Pattern

- Examples
  - Operating systems and user applications
  - Layered architecture for .NET

Example


Shared Types
Entities (BE)
Enums
Presentation
User Interfaces (UI)
Controllers (UIC)
External Systems
Frameworks
Service
Contracts
Services (SI)
Workflow Services (WFS)
Security
Validation
Exception
Logging
Configuration
Business
Business Components (BC)
Workflows (WF)
Activities (WFA)
Data
Data Access Components (DAC)
Data Agents (DA)
Other Reusable Blocks
Database
Services & Data Sources

Example


user and other system programs
GUI
batch
command line
user interfaces
system calls
program execution
I/O operations
file systems
communication
resource allocation
accounting
error detection
protection and security
services
operating system
hardware

Architecture styles/patterns

- Model-View-Controller Pattern
- Layered Pattern

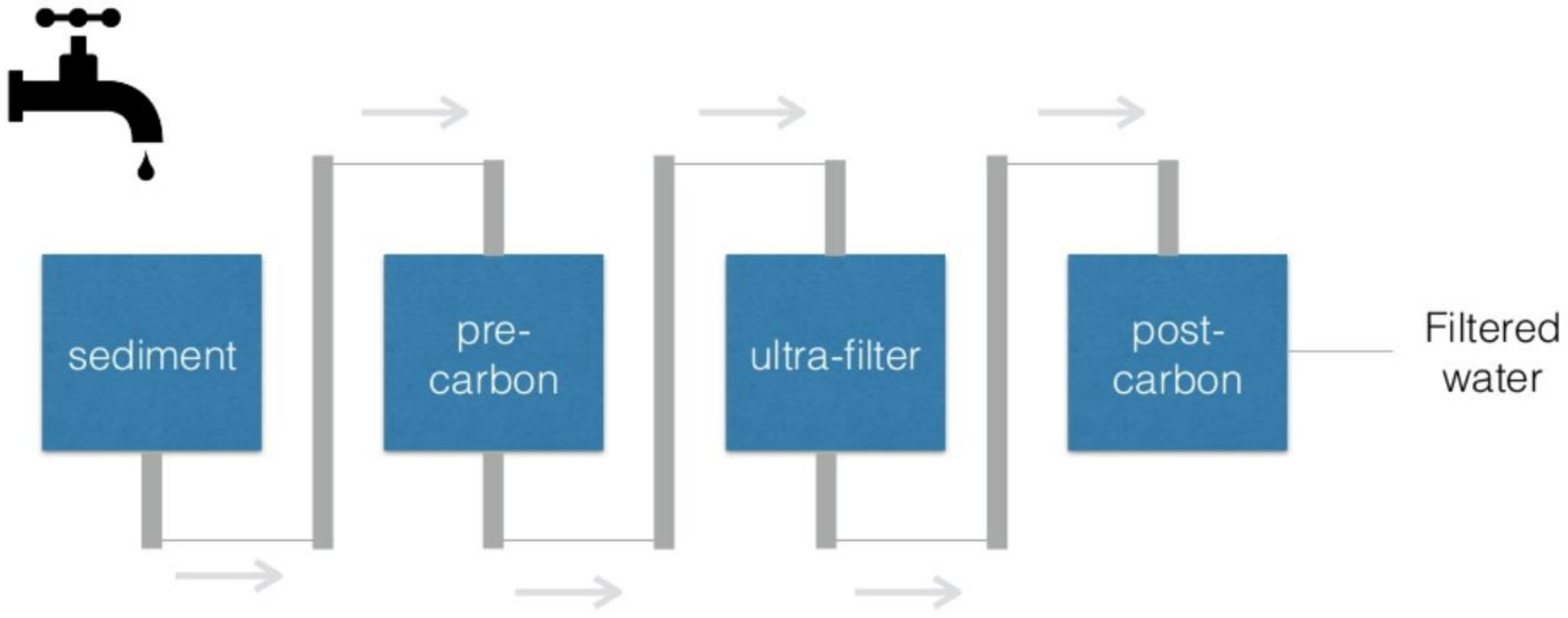
Architecture styles/patterns

- Model-View-Controller Pattern
- Layered Pattern
- Data-flow Pattern

Data-flow pattern

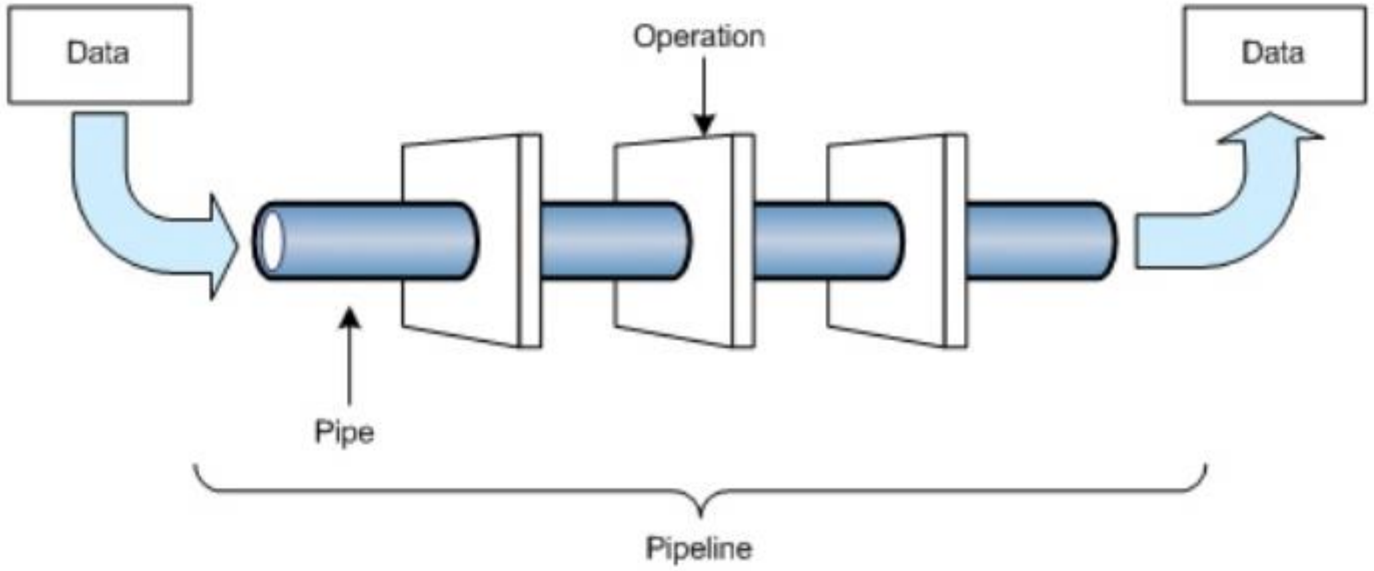
- Pipe-and-filter

Data-flow pattern



Data-flow pattern

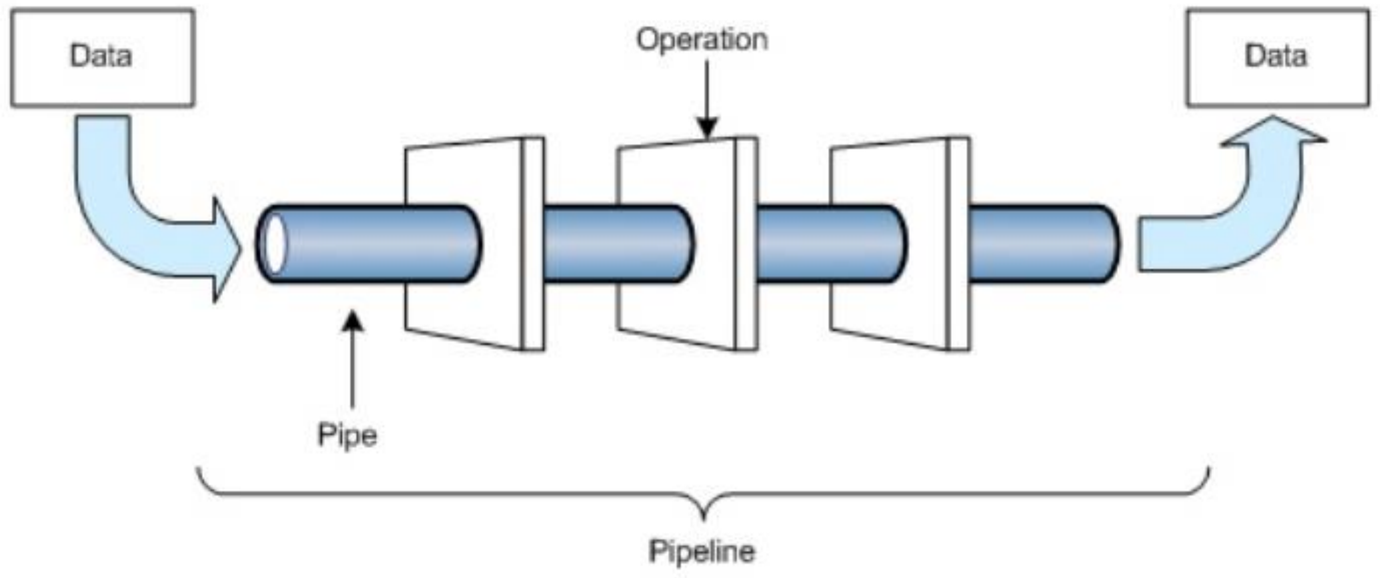
- Pipe-and-filter



Data-flow pattern

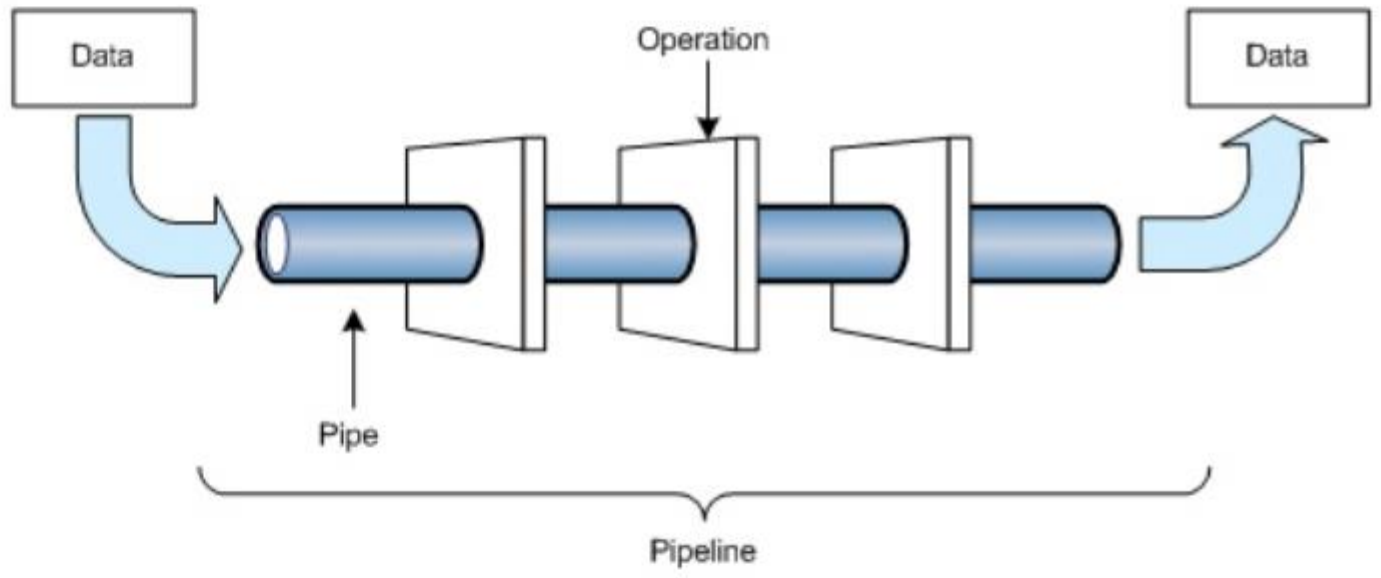
- Pipe-and-filter

✓ Flexibility by filter exchange
✓ Flexibility by recombination
✓ Reuse of filter components
✓ Rapid prototyping of pipelines



Data-flow pattern

- Pipe-and-filter

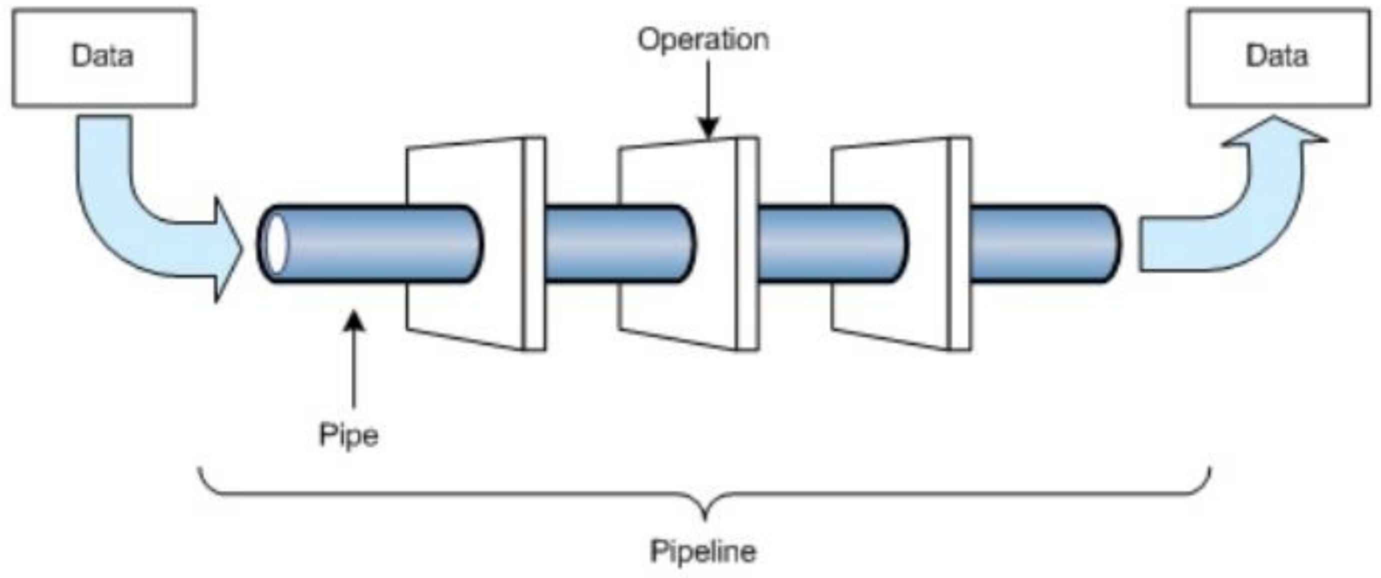


- ✓ Flexibility by filter exchange
- ✓ Flexibility by recombination
- ✓ Reuse of filter components
- ✓ Rapid prototyping of pipelines

- ▪ Sharing state information is expensive or inflexible
- ▪ Efficiency gain by parallel processing is often an illusion
- ▪ Data transformation overhead
- ▪ Difficult to handle errors

Data-flow pattern

- Pipe-and-filter



Examples

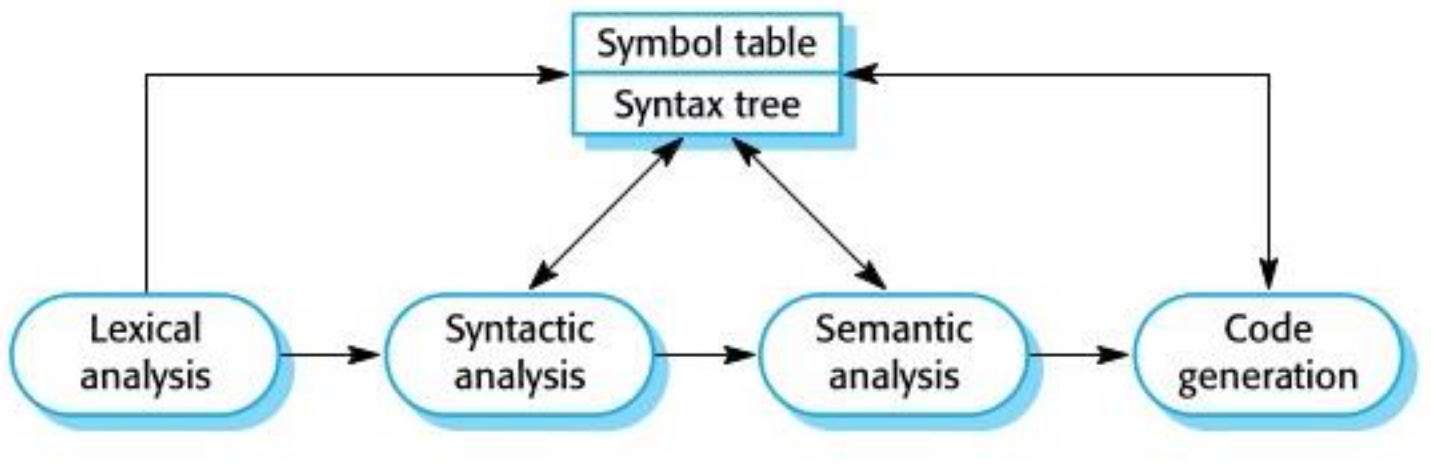
- Compiler

Data-flow pattern

- Pipe-and-filter

Examples

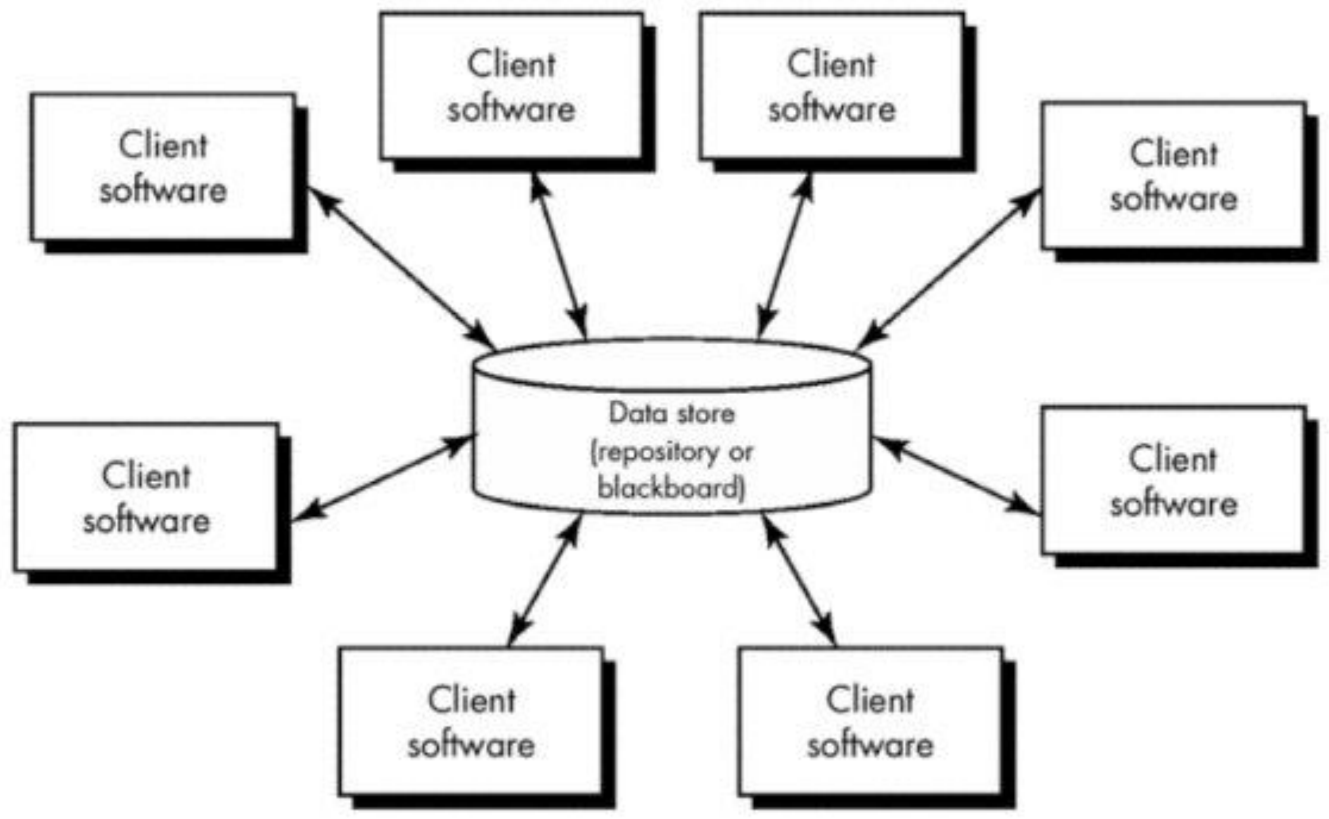
- Compiler



Architecture styles/patterns

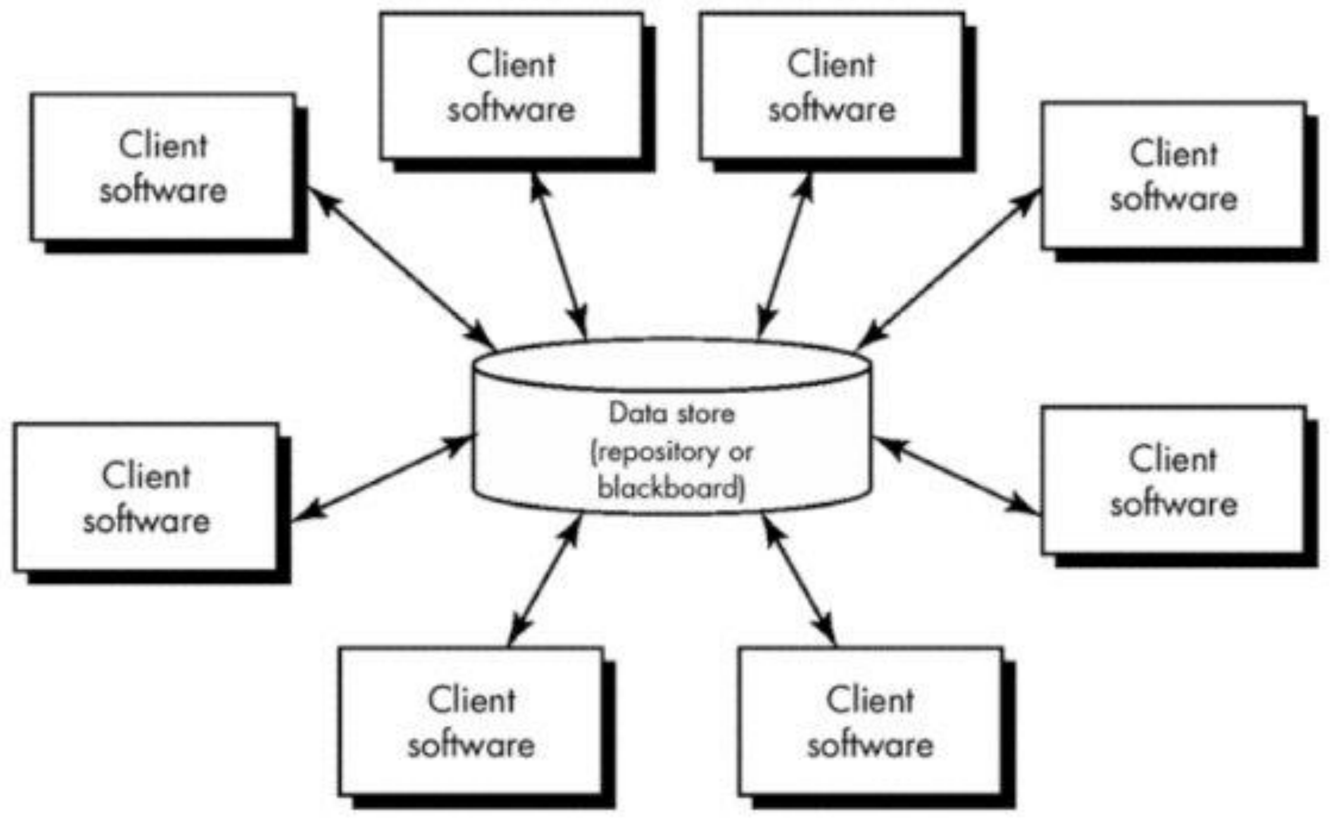
- Model-View-Controller Pattern
- Layered Pattern
- Data-flow Pattern
- Data-centered Pattern

Data-centered Pattern



A centralized data store
A number of clients
And communication between them

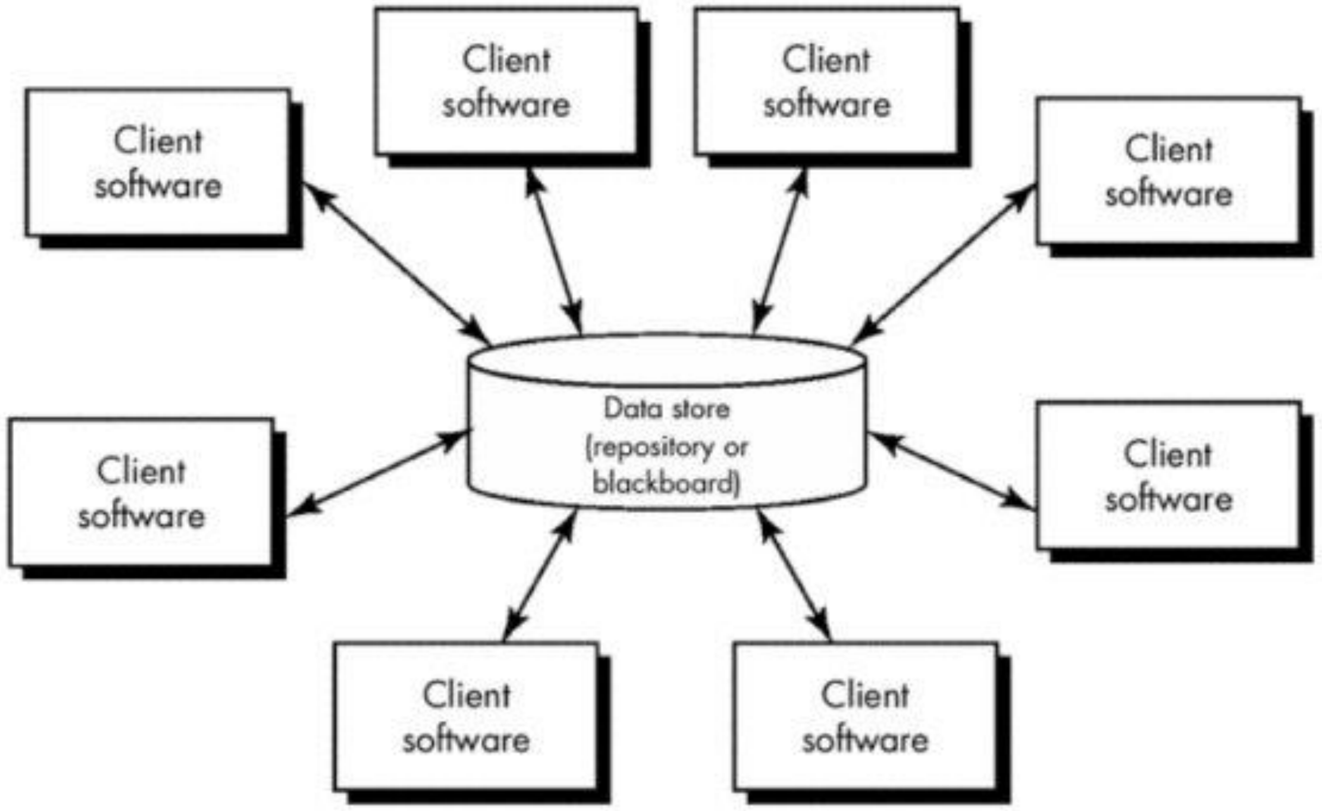
Data-centered Pattern



Goal: integrating the data

- Refers to the systems
  - Access and update of a widely accessed data store occur

Data-centered Pattern

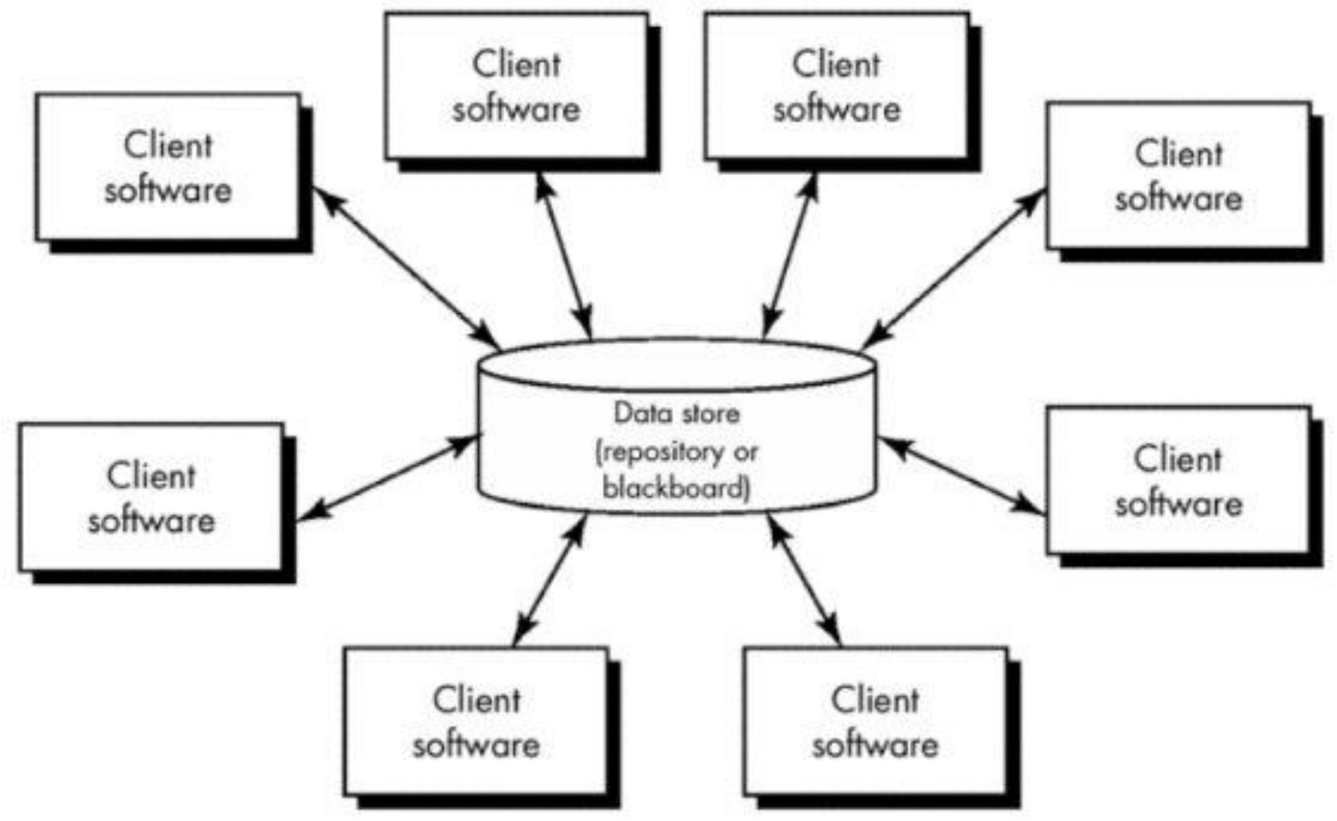


Goal: integrating the data

- Refers to the systems
  - Access and update of a widely accessed data store occur

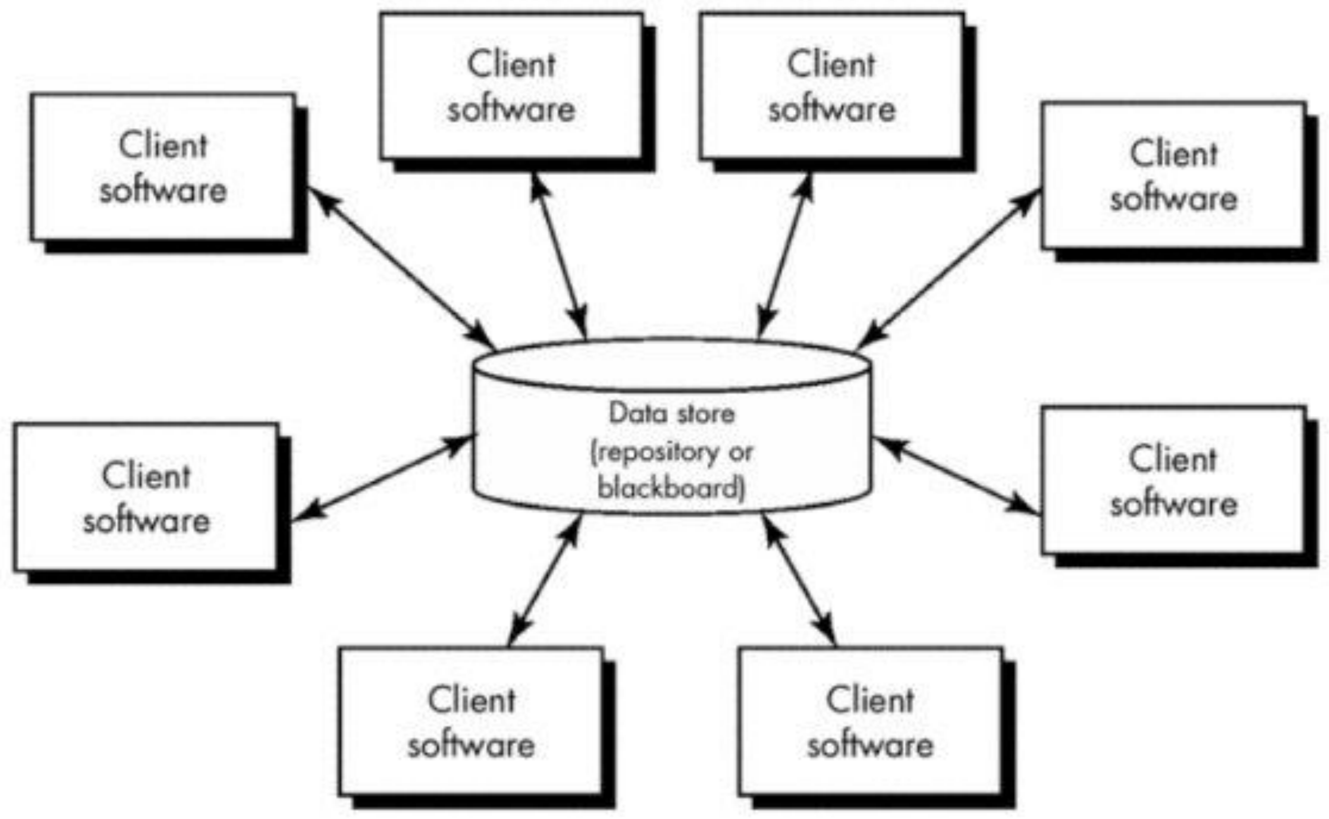
- The data center is independent of the clients.
- The clients are relatively independent of each other so that they can be added, removed, or changed in functionality.

Data-centered Pattern



Weakness?

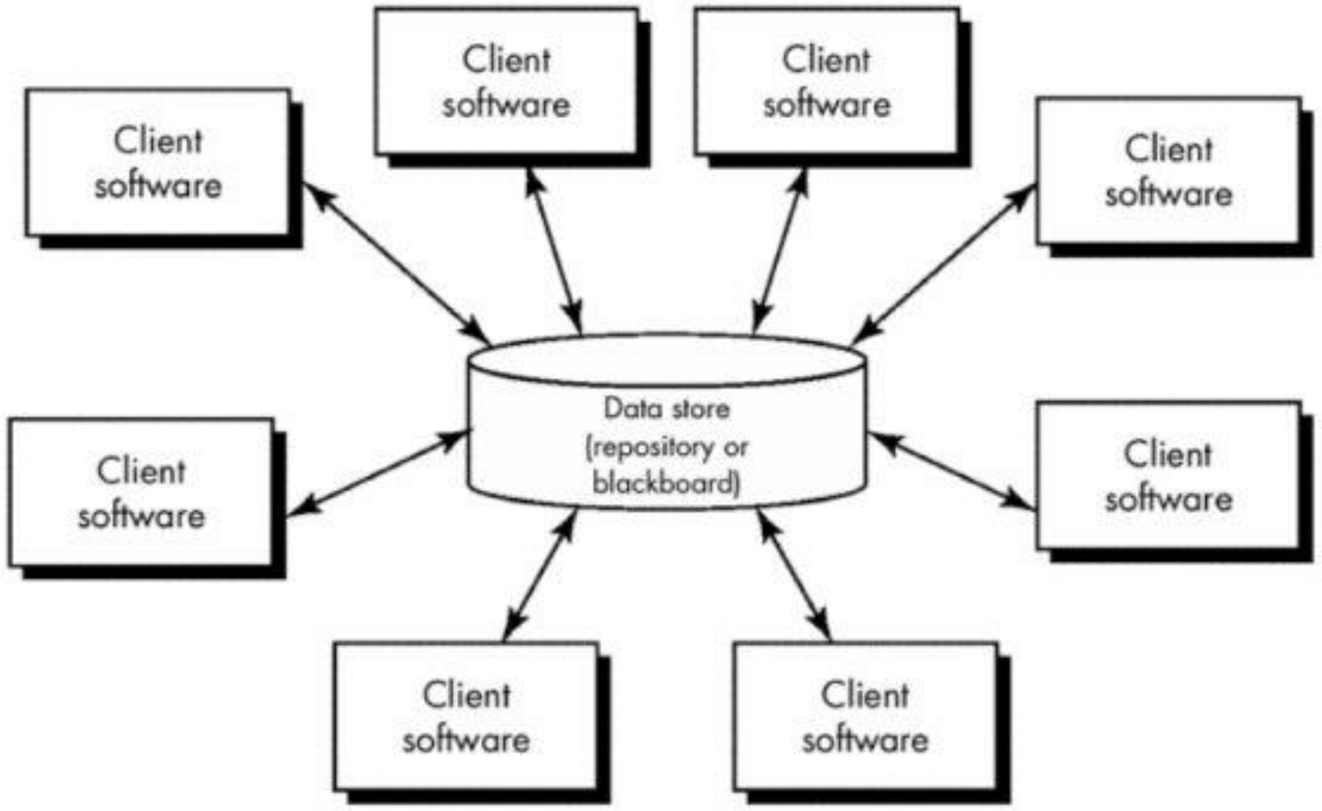
Data-centered Pattern



Weakness?

- The data repository is the single-point-of-failure, performance bottleneck

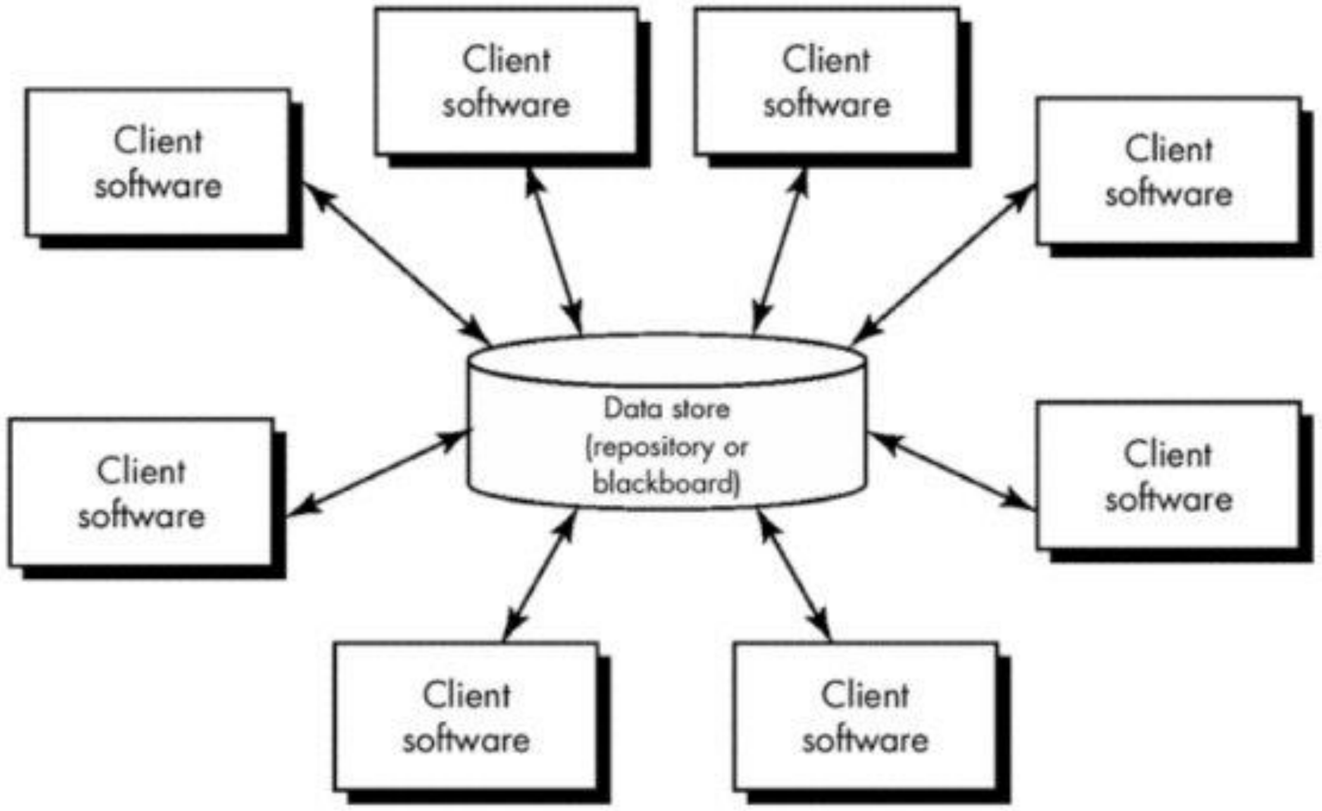
Data-centered Pattern



Weakness?

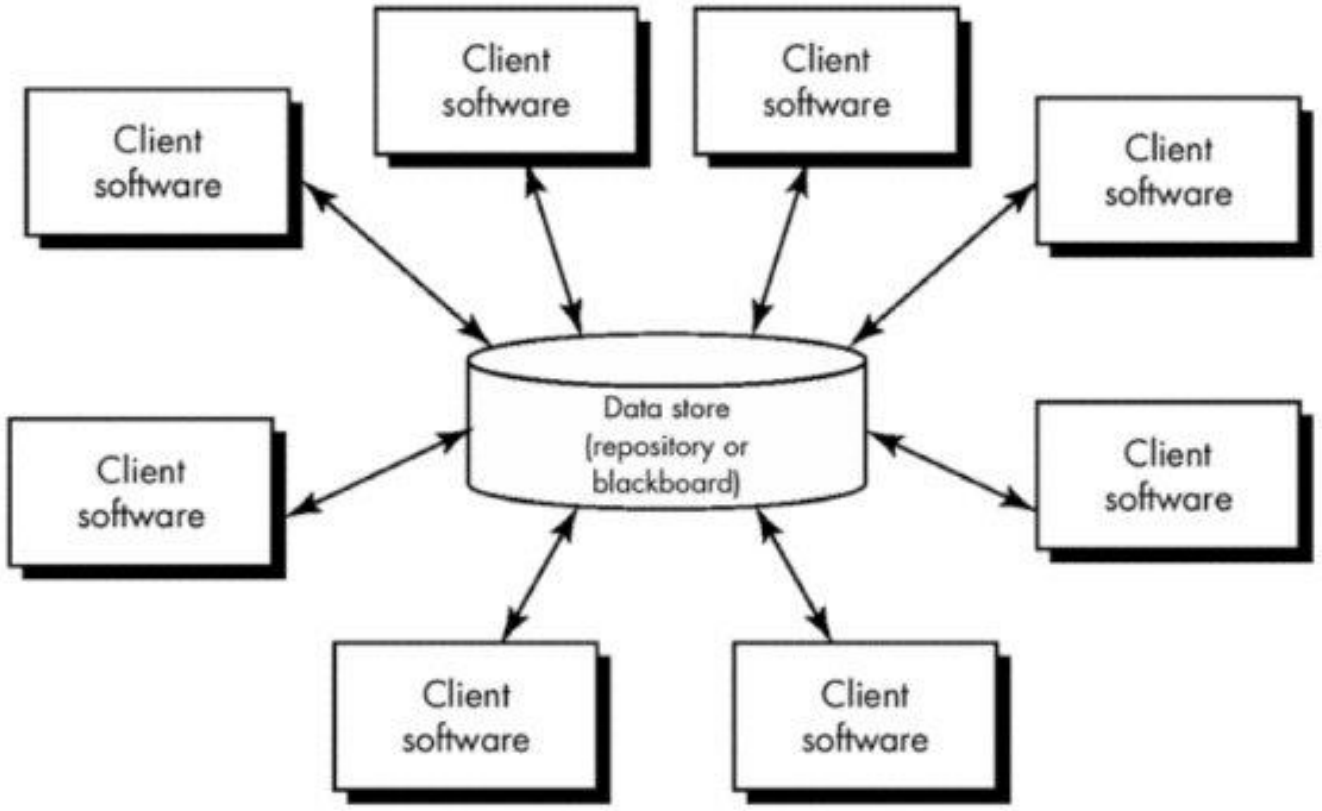
- The data repository is the single-point-of-failure, performance bottleneck
- Slow for different components to interact with each other

Data-centered Pattern



Example?

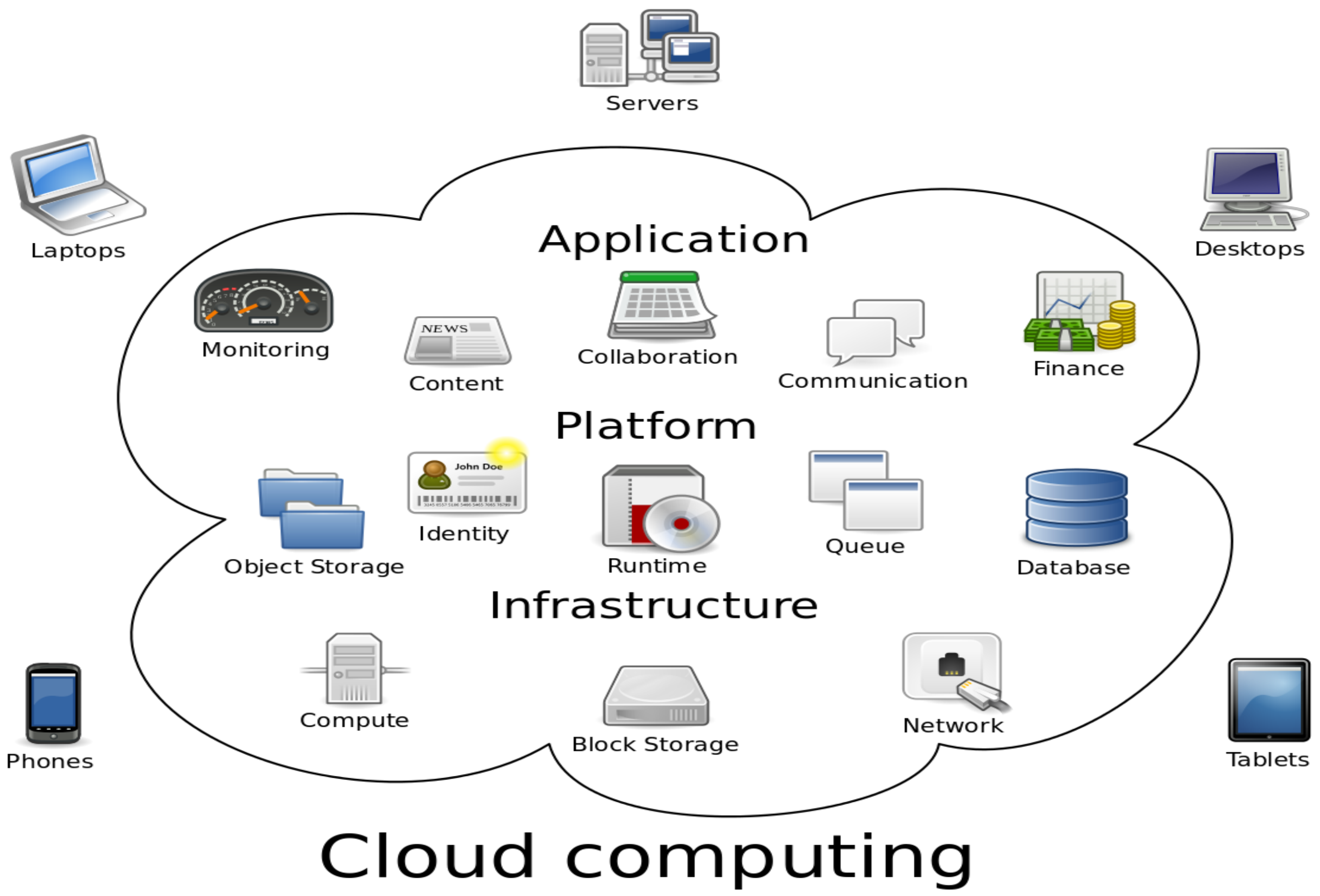
Data-centered Pattern



Example?

- Clouding computing

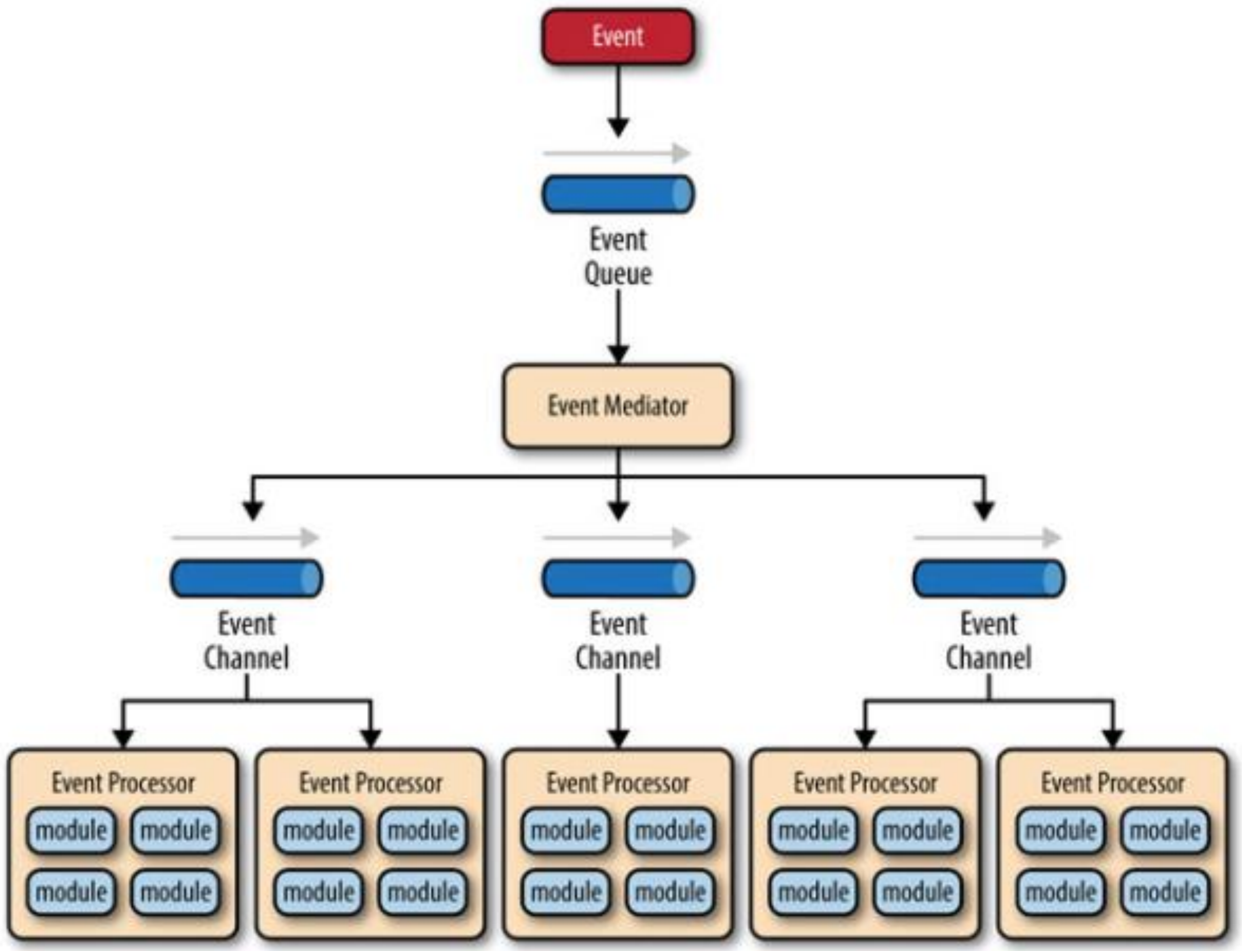
Data-centered Pattern



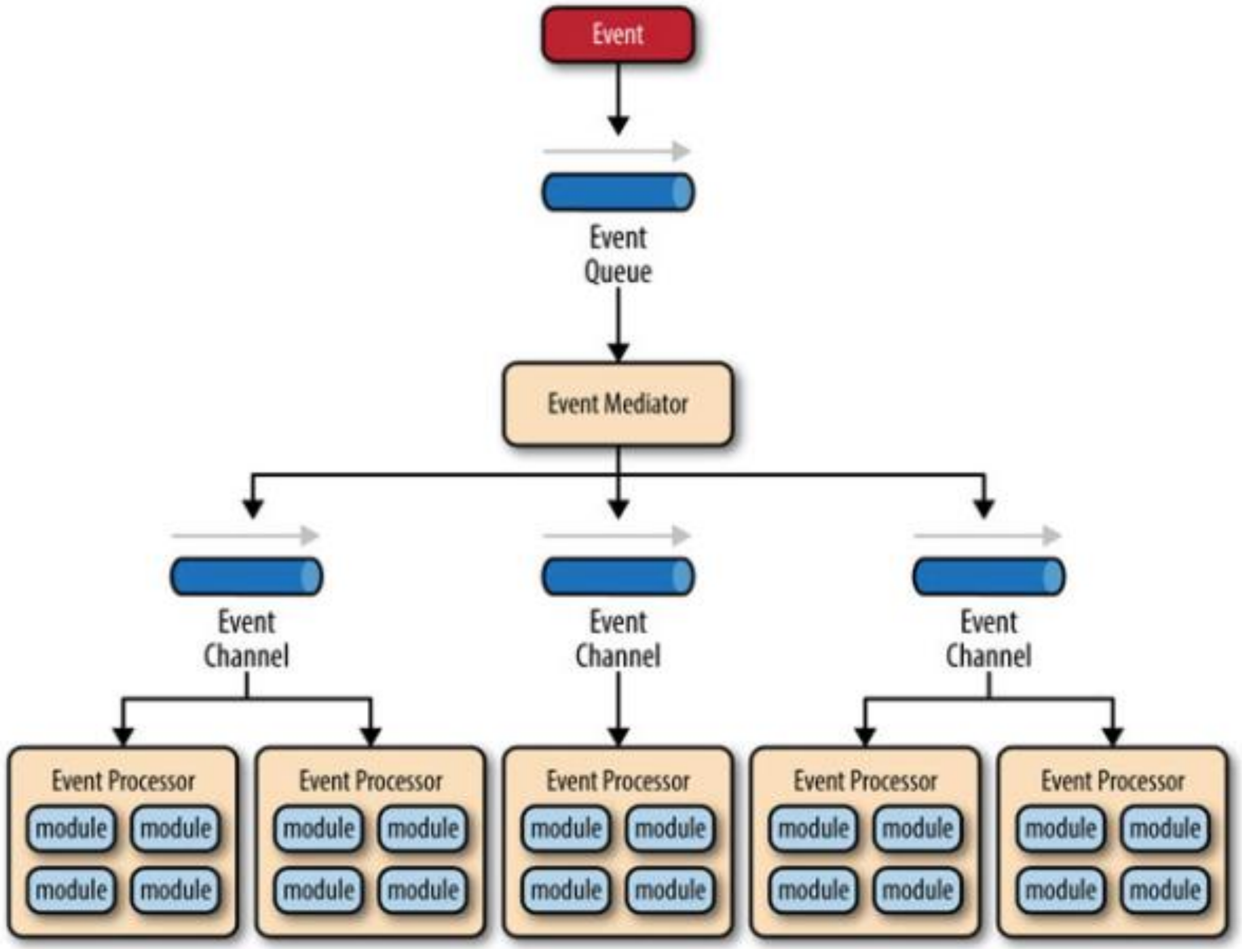
Architecture styles/patterns

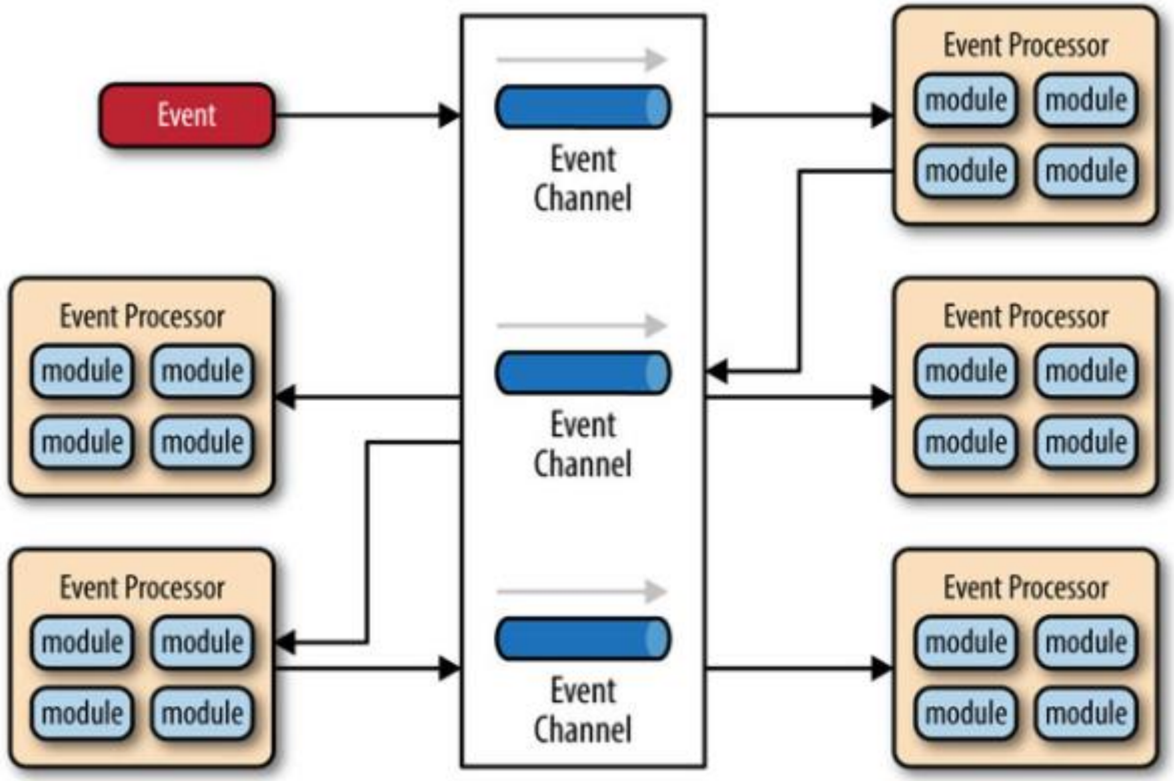
- Model-View-Controller Pattern
- Layered Pattern
- Data-flow Pattern
- Data-centered Pattern
- Event driven Pattern

Event-driven Pattern



Event-driven Pattern





Event-driven Pattern

- Real-time systems are often event-driven, with minimal data processing.
  - e.g., a landline phone switching system responds to events such as ‘receiver off hook’ by generating a dial tone
- Facilitates the information flow between producing and consuming systems
- The flow of the program is controlled by user-generated events.

Architecture styles/patterns

- Model-View-Controller Pattern
- Layered Pattern
- Data-flow Pattern
- Data-centered Pattern
- Event driven Pattern

Architectural design assessment

- Cohesion(within component)
  - The more the better
- Coupling(across components)
  - The less the better

How do you define the architecture of your project?