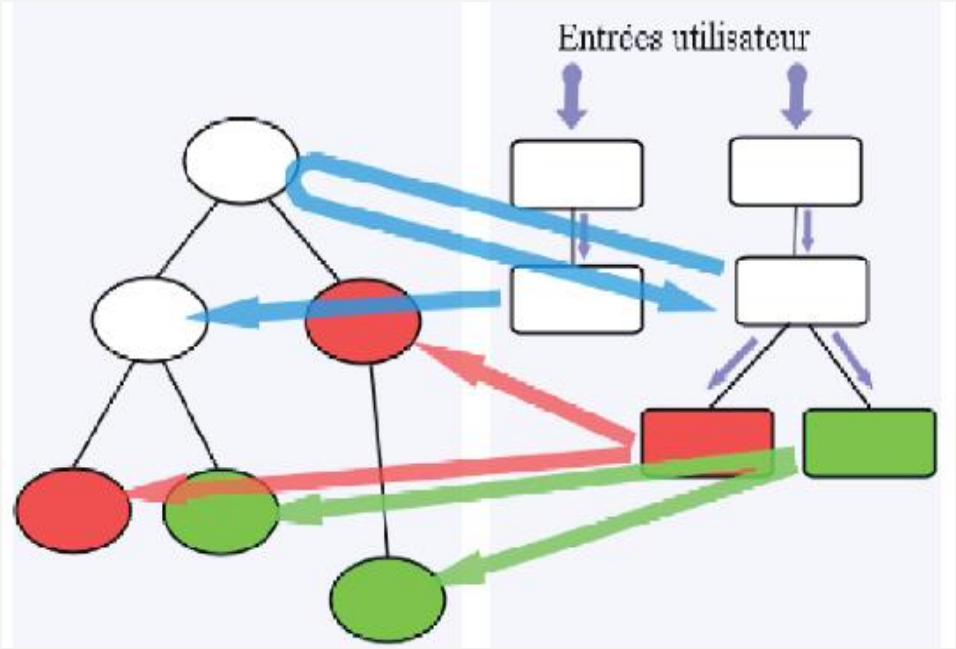
University of Mila

3rd Year Computer Science



HUMAN – MACHINE INTERACTION

Chapter 3: Models & Architectures

PLAN

III.1- The Dialog Controller (definition & role).

III.2- Presentation of the Seeheim Model

III.3- Presentation of the PAC Model

III.4- Presentation of the MVC Model

INTRODUCTION

Observation:

• Difficult IHM design, consequently iterative (iterativity implies software modifiability)

• Increasing complexity and size of IHMs

• Imperfect IHM development tools (toolkits, application skeletons, interface generators)

- **Consequence:**
  We need a framework for thought.
- Architecture is abstract; it describes components and their relationships without presuming their implementation. There are proven architectures that serve as reference models.

ARCHITECTURE MODELS

Define the software organization of an interactive system.

- **Basic Principle:**
  The separation between the **functional core**, which implements concepts specific to a particular application domain, and the **interface**, which presents these concepts to the user and allows them to manipulate them.

- **Role:**
  It should theoretically allow modifying the interface without affecting the functional core and vice versa, though in reality, this is difficult to achieve for the entire interface.
  In practice and in most cases, only a part of the interface is truly separated from the functional core.
- It allows for structuring an interactive system, thus providing better modularity (essential given the iterative method of interface construction). It facilitates software reuse of interactive system components, its maintenance, and its evolution.

WHY USE AN ARCHITECTURE?

- Organize the code (tidiness)
  - Simplify (divide and conquer)
  - Organize work
  - – Iterative
  - – Parallel (merging)
  - Modify (a part)
  - Reuse

Notions: modularity, scalability, flexibility

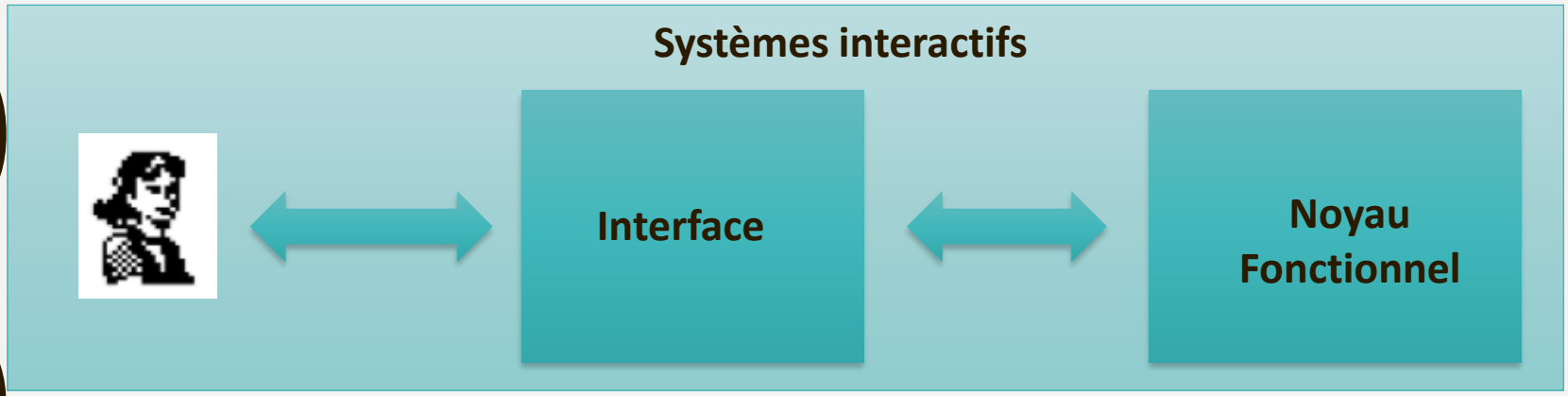
IHM AND ARCHITECTURE

- Possible separation
  - Code for IHM
  - Code for Functions, Objects, Services

- **Objective: Avoid having to change everything if the functional part or the IHM part is modified.**

INTERACTIVE SYSTEMS

All models start from the principle: an interactive system has an **interface** part and a **pure application** part.
The latter is often called the **functional core**.



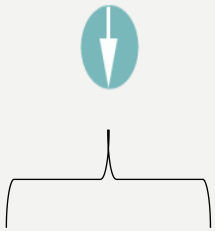
SOFTWARE ARCHITECTURES

Most models generally identify three types of elements:
a "user side" (presentations, views),
a "functional core side" (interfaces of the functional core, abstractions, models),
and articulating elements (controllers, adapters).

DEFINITION: SOFTWARE ARCHITECTURE

- Describes, in a symbolic and schematic manner, the different elements of one or more computer systems, their interrelationships, and their interactions.

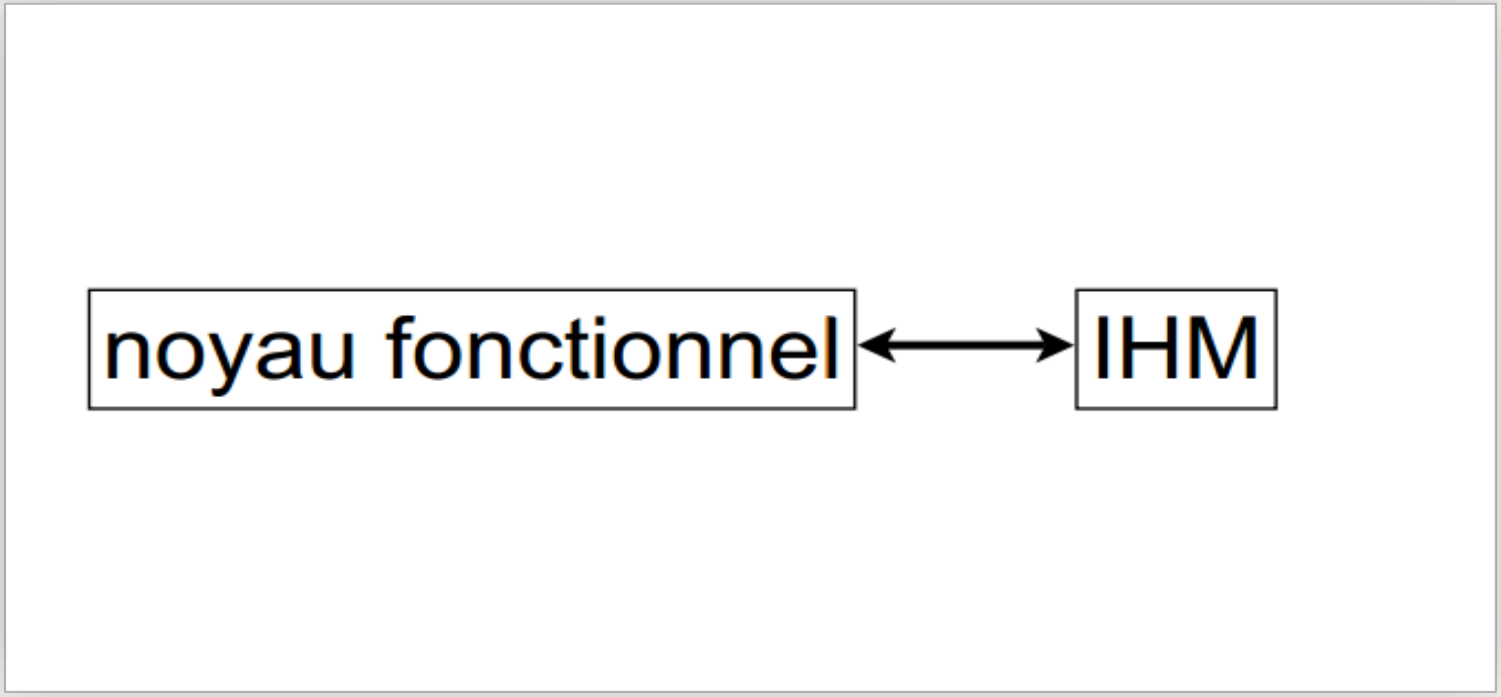
Models:



Layered Model

Agent Model

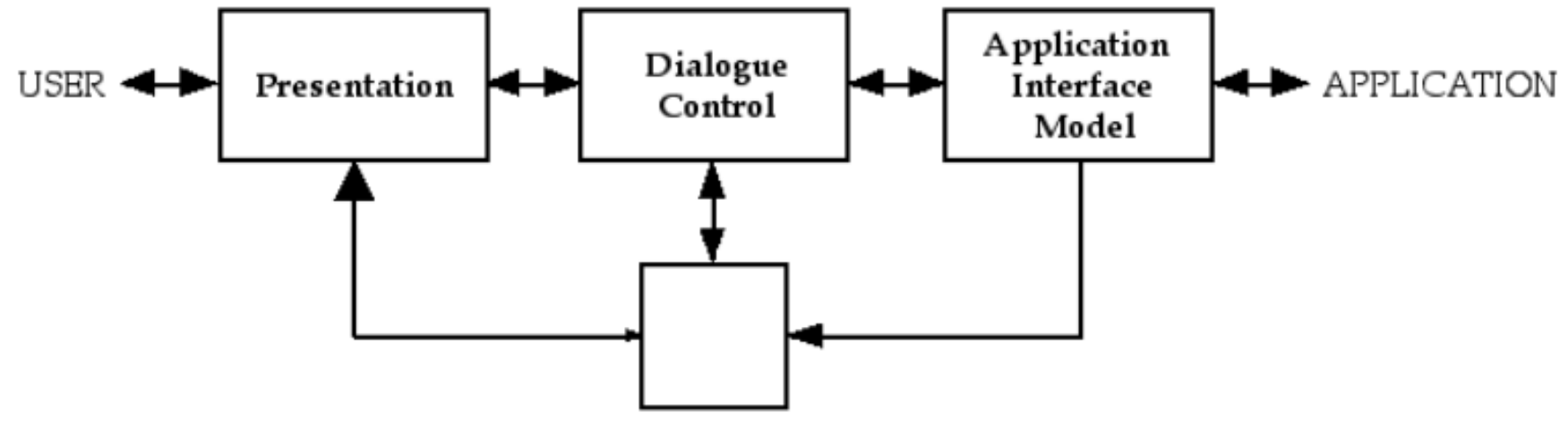
FUNDAMENTAL ARCHITECTURE



THE SEEHEIM MODEL

LAYERED MODEL

PRESENTATION OF THE SEEHEIM MODEL



PRESENTATION OF THE SEEHEIM MODEL

- **The Presentation Component:**
  The effective production of outputs (display, sound production...) based on information received from the dialog controller,
  The reception and (first-level) processing of input events induced by user actions, then transmitting the resulting information to the dialog controller.
- **The Dialog Controller:**
  Where the structure of the dialogue between the user and the application is determined.
  The dialog controller is responsible for the sequencing of operations at the level of user actions and calls to domain functions.
  It acts as a bridge between the concrete objects of the interface (presentation component) and the concepts of the domain (functional core).

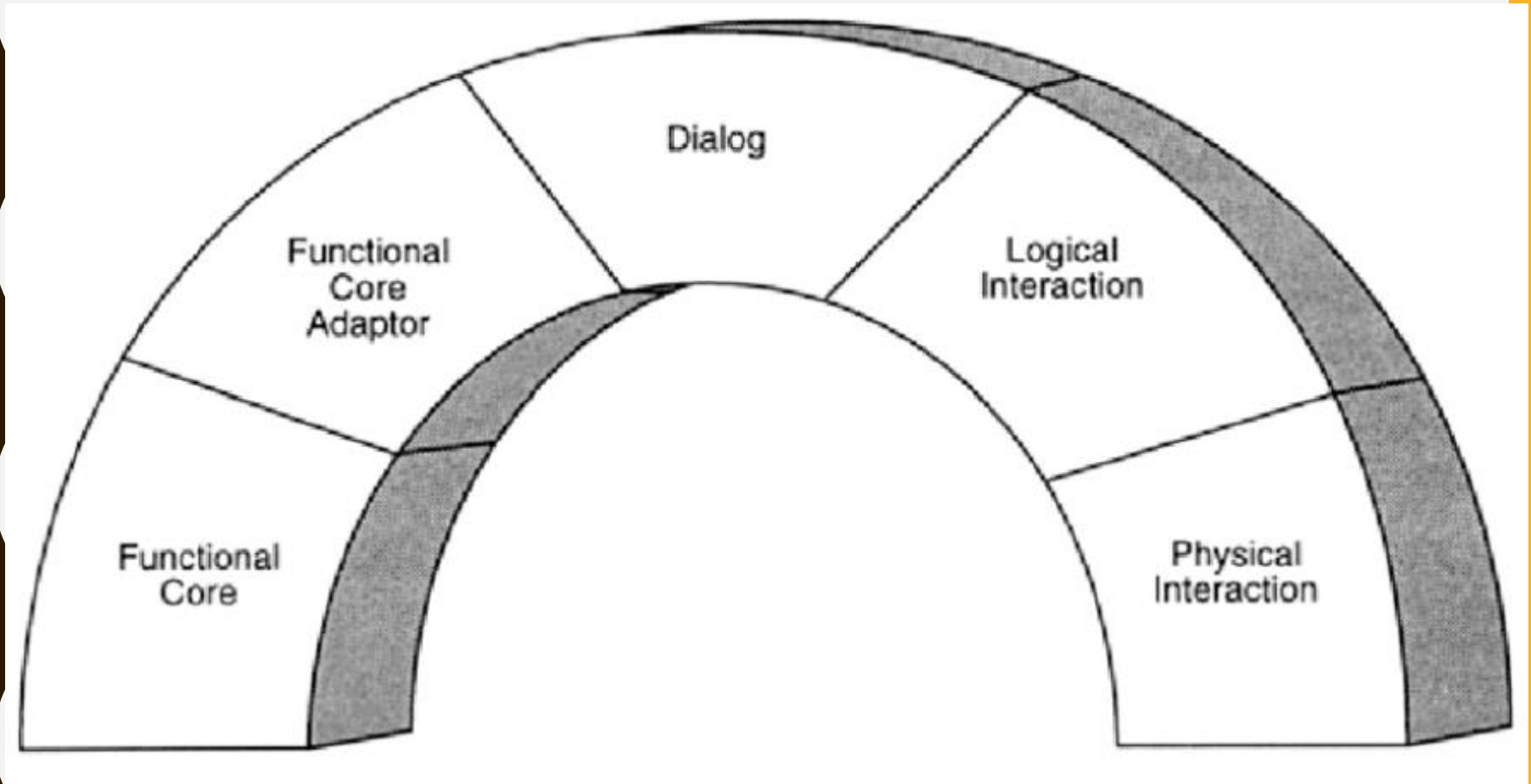
PRESENTATION OF THE SEEHEIM MODEL

- **The Functional Core Interface:**
  This component serves as a gateway between the functional core and the dialog controller. It adjusts the differences between the formalisms used by these two components.
  It generally contains a description of the domain entities that are directly linked to the interface, as well as a description of the functional core procedures that can be invoked by the dialog controller.

THE SEEHEIM MODEL [PFAFF, 1985]

- **The Functional Core:** manipulates the objects of the application domain.
- **The Functional Core Interface:** describes the application's semantics from the user interface's point of view.
- **The Presentation:** Defines the system's behavior as perceived and manipulated by the user.
- **The Dialog Controller:** acts as a mediator between the functional core interface and the presentation.

THE ARCH MODEL



THE ARCH MODEL [1992]

The functional core and the interaction component (usually a toolkit) form the feet of the arch, because in most cases these two components exist before the development of the interface itself and thus form the starting base.

THE ARCH MODEL [1992]

Arch is a refined version of Seeheim that accounts for the emergence of toolkits.

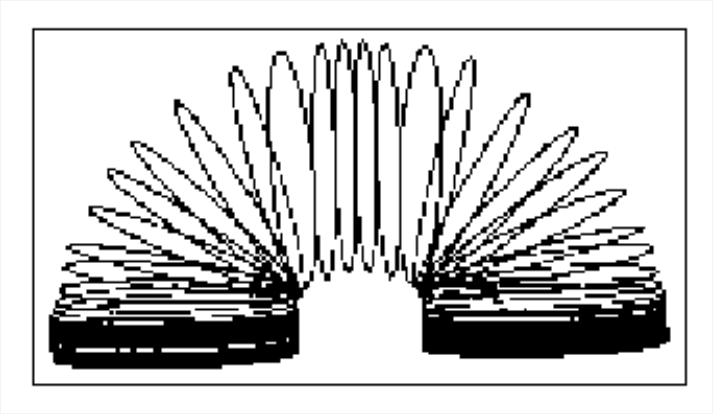
The feet of the arch represent the pre-existing elements: the functional core on one side and the interface toolkit on the other.

The presentation and the functional core adapter allow for adaptation between the 3 other components.

These adaptations are not always necessary: the Slinky meta-model allows for their removal or their fusion into the other elements of the arch.

THE SLINKY META-MODEL

- The term SLINKYTM comes from a flexible toy that, once set in motion, sees its center of gravity shift due to a change in the distribution of its mass.
- This metaphor is used in the ARCH model to represent the modification of the distribution of functionalities across the different components of the model, from one instance to another.



AGENT MODEL

AGENT REFERENCE MODELS

Principle: An interactive system consists of a collection of specialized computing units (agents).

An Agent:

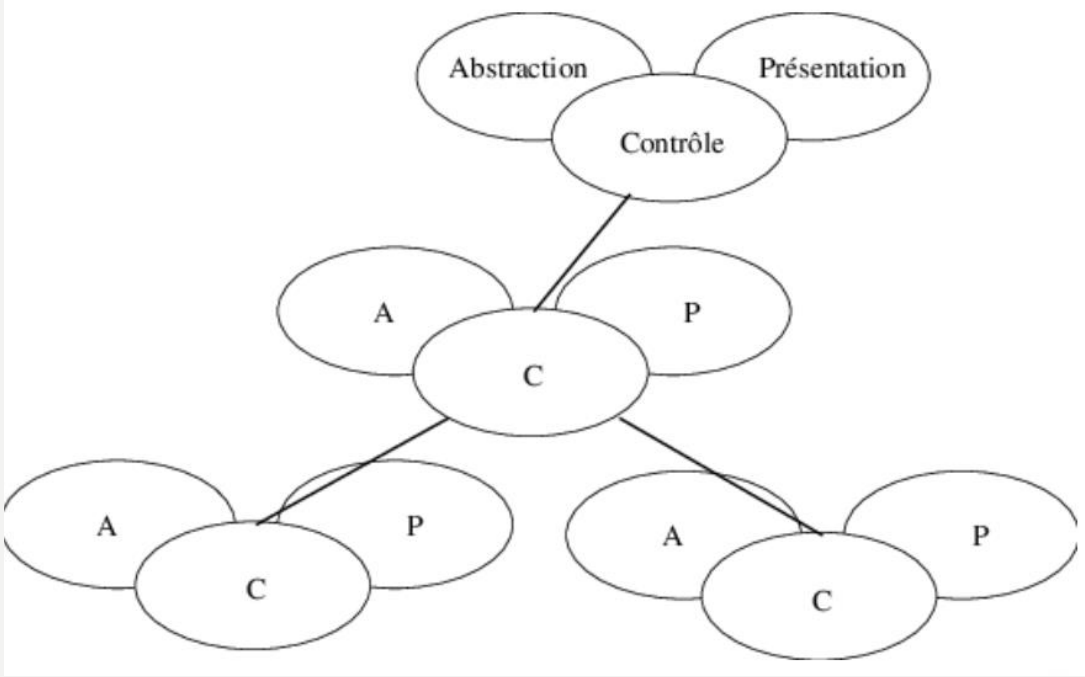
- has a state;
- has expertise; and
- is capable of emitting and reacting to events.

An **interactor** is an agent in direct contact with the user.

PAC MODEL

PAC MODEL [COUTAZ, 1987]

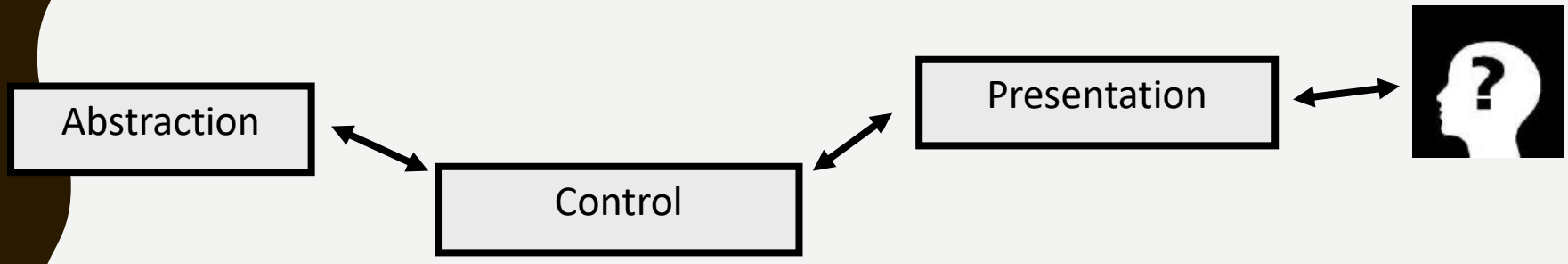
- It structures the interactive application recursively in the form of a hierarchy of agents, each comprising three facets.



PRESENTATION OF THE PAC MODEL

A PAC agent is composed of three facets implemented by objects:

- The **Abstraction** (the model in MVC);
- The **Presentation** (the view and controller in MVC); and
- The **Control** which expresses the dependencies between abstraction and presentation and manages exchanges with other agents.



PAC MODEL [COUTAZ, 1987]

- The **Control** notifies the model when user manipulations of the **presentation** require it; and notifies the **presentation** when **modifications** to the model require it. The **Control** notifies the control facets of other PAC agents in the hierarchy if needed.

MVC MODEL

PRESENTATION OF THE MVC MODEL

- MVC is: A design pattern (a standardized solution to a problem, independent of programming languages) A software architecture (a way of structuring an application or a set of software)

  Introduced in 1979 by Trygve Reenskaug.

  Very strongly linked to the concepts of object-oriented programming (Smalltalk).

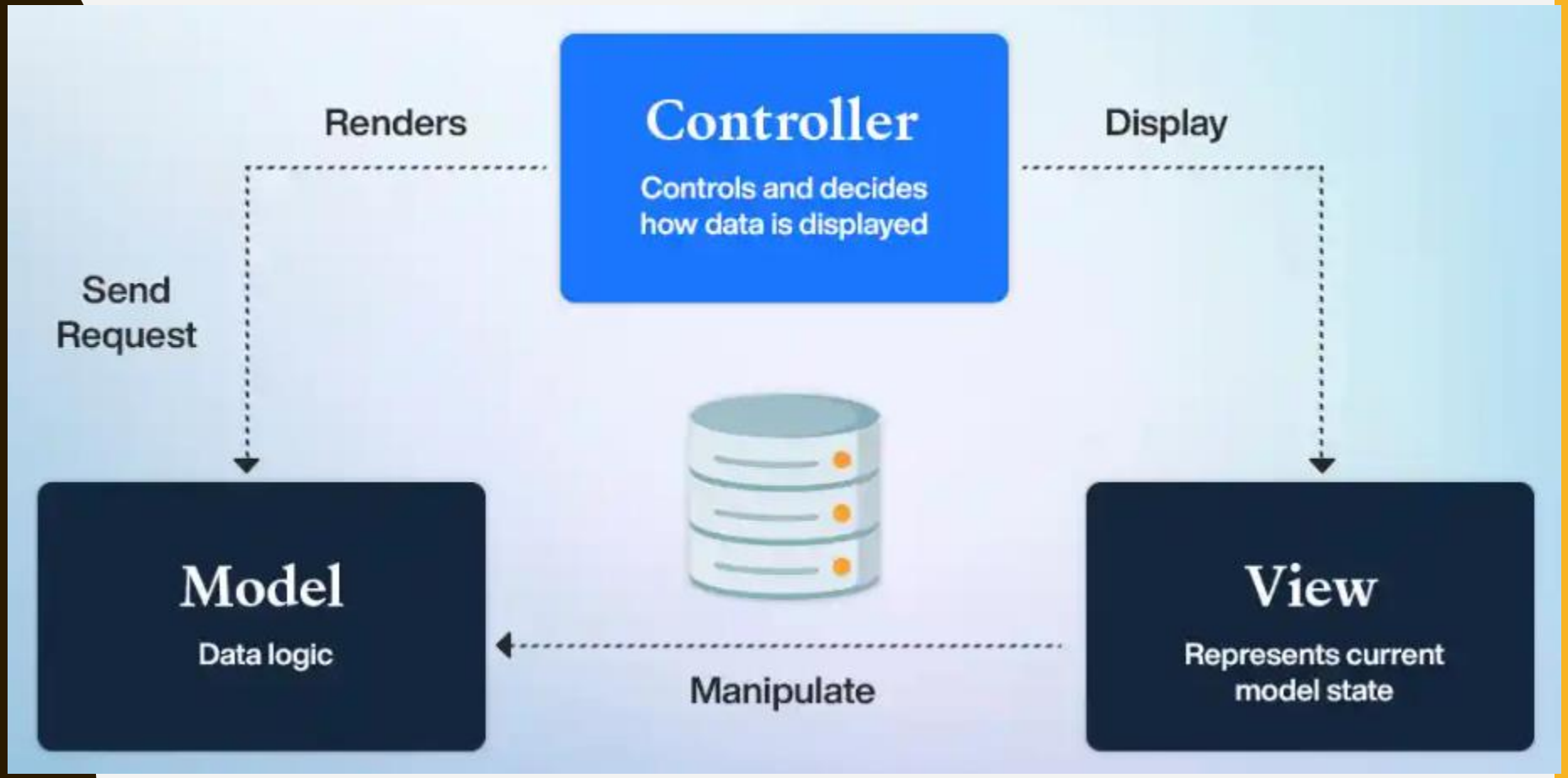
PRESENTATION OF THE MVC MODEL

- Model-View-Controller is the multi-agent model used by Smalltalk [GOLDBERG 84].
- Each of the three components of the MVC triad is a full-fledged object.

  Cause: difficulties in designing highly interactive applications.

  Response: modularization.**Principe de base**

MVC ARCHITECTURE



PRESENTATION OF THE MVC MODEL

- **Basic Principle**

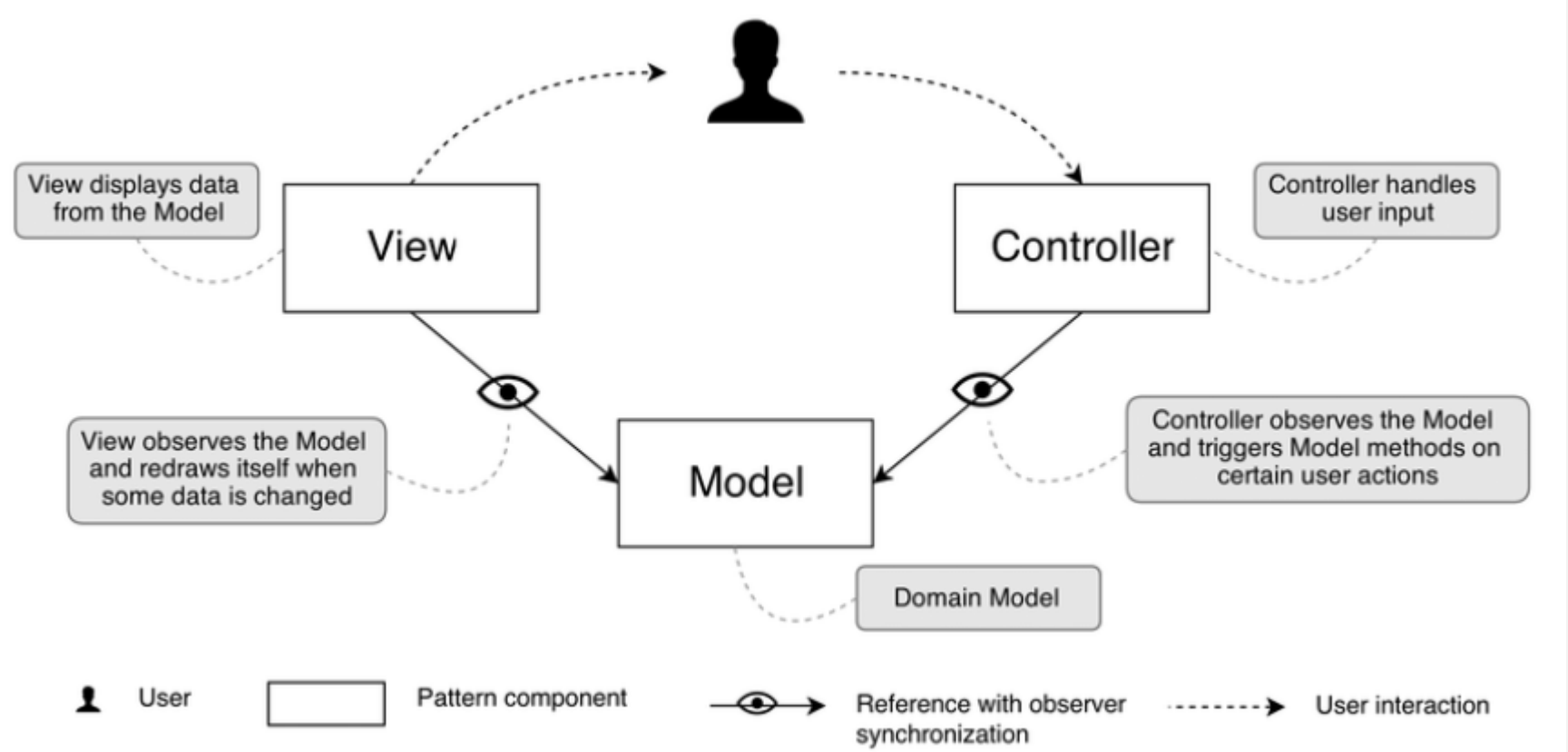
  – **Model:** modeling (data and behavior), maintains its state and notifies of its modifications.

  – **View:** presents information to users using data from the models.

  – **Controller:** handles interaction with the user. The controller listens to the user and requests modifications from the model.

MVC MODEL 1987 (SMALLTALK)

MVC 1987 (Smalltalk): an interactive system as a set of agents, each agent having three facets.



MVC: MODEL STRUCTURE

Organize, structure an interactive application by separating:

- The data and their processing: **The Model**
- The representation of the data: **The View**
- The behavior of the application: **The Controller**

The model, view, and controller communicate by exchanging messages.

MVC: THE MODEL

- Functional Core of the application
- Represents the data
- Provides access to the data
- Provides applicable data processing
- Exposes the application's functionalities

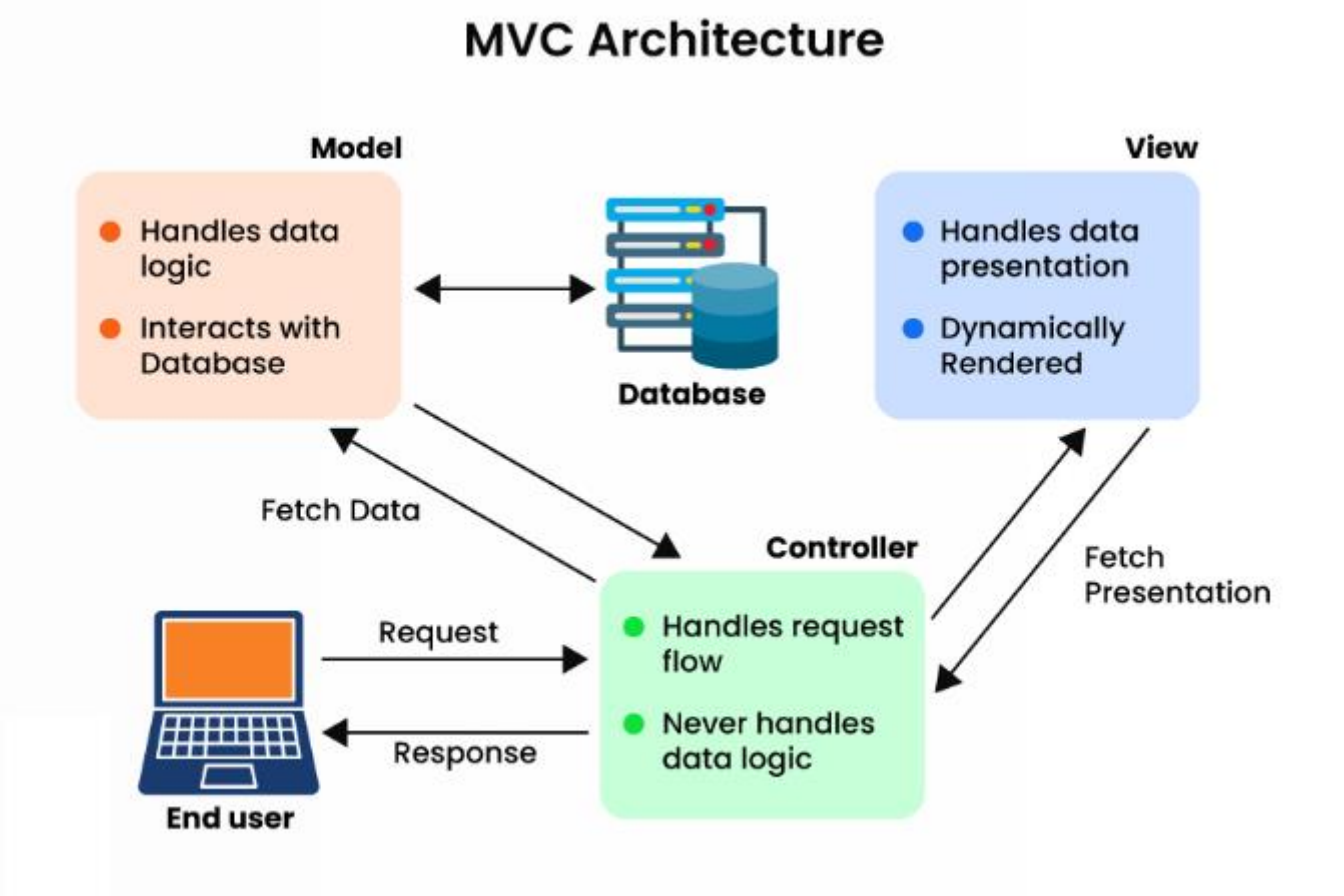
MVC: THE VIEW

- **Application outputs**
  - Represents the (or a) representation of the model's data
  - Ensures consistency between the representation it provides and the state of the model / the application context

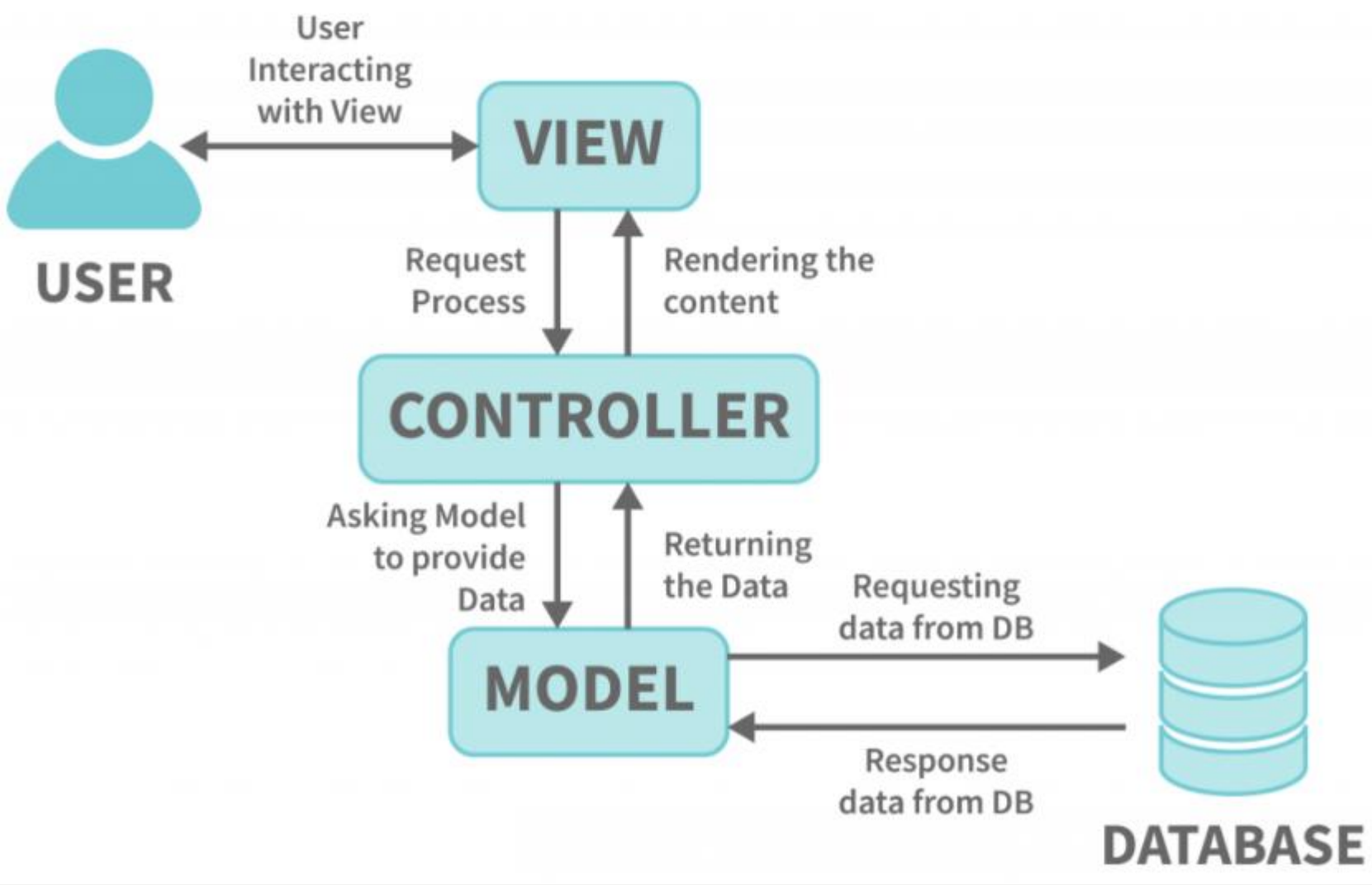
MVC: THE CONTROLLER

- Behavior and management of application inputs
  - Represents the behavior of the application in response to user actions
  - Provides the translation of user actions into actions on the model
  - Provides the appropriate view based on user actions and model reactions

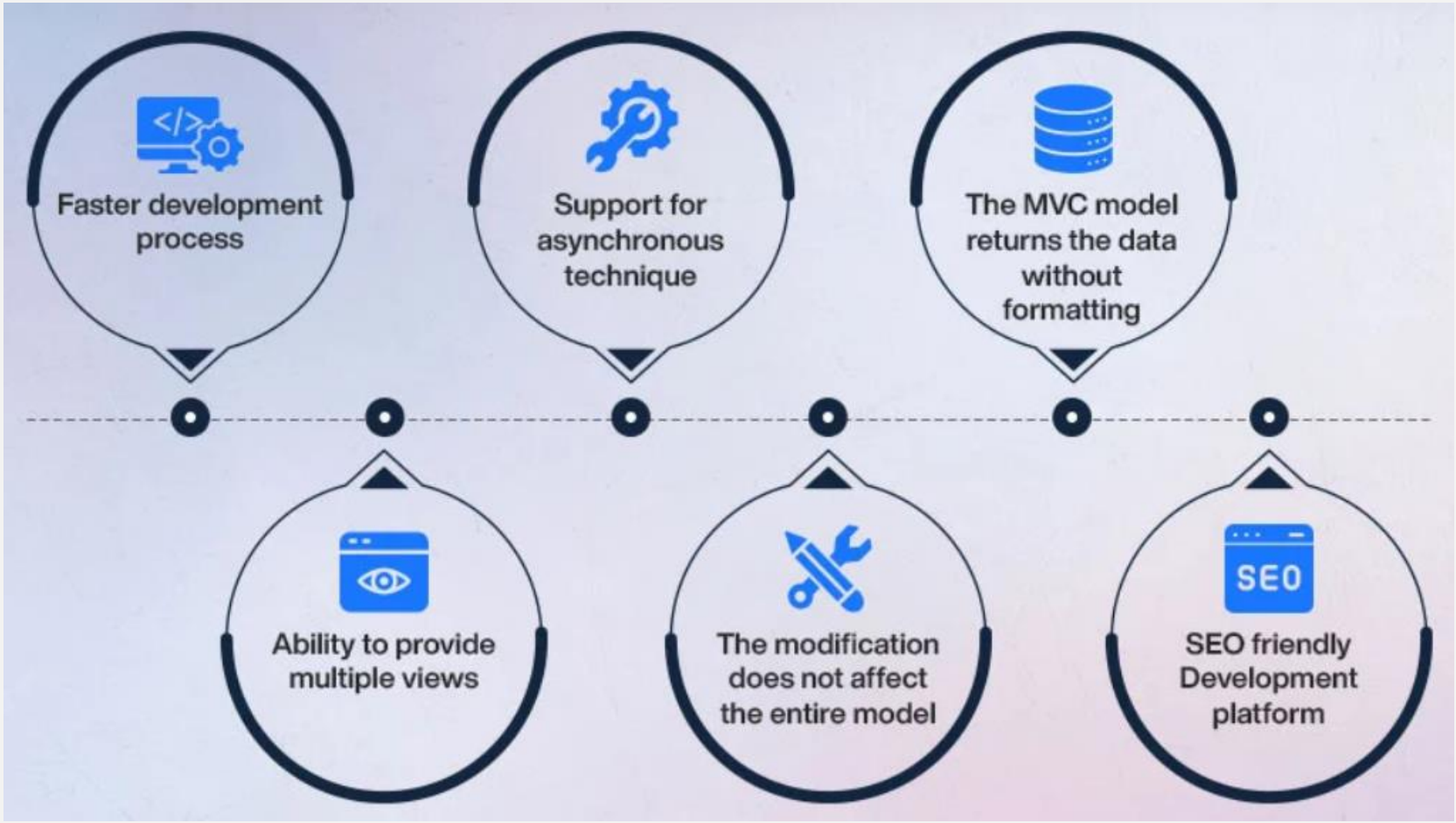
COMMUNICATION BETWEEN MVC COMPONENTS



MVC: SUMMARY


User
Interacting
with View
VIEW
USER
Request
Process
Rendering the
content
CONTROLLER
Asking Model
to provide
Data
Returning
the Data
Requesting
data from DB
MODEL
Response
data from DB
DATABASE

ADVANTAGES OF USING MVC FRAMEWORK



ADVANTAGES AND DISADVANTAGES

Advantages

- 'Clean' application structure
- Independence between 'data', 'representation', and 'behaviors'
- Modular and reusable

Disadvantages

- Complex setup for large applications
- Potentially too many updates / 'Spaghetti' code
- Controller and View often remain strongly linked to the Model
- Complexity of communication between components

CONCLUSION

- Separation of concerns is one of the most attractive concepts of the MVC pattern. The complexity of modern web applications can make it difficult to make changes. When the frontend and backend are separated, the application is scalable, maintainable, and easy to expand.

THANK YOU