Appreciation for Software Architecture

SWEN-261
Introduction to Software Engineering

Department of Software Engineering
Rochester Institute of Technology

Software Engineering
Rochester Institute of Technology

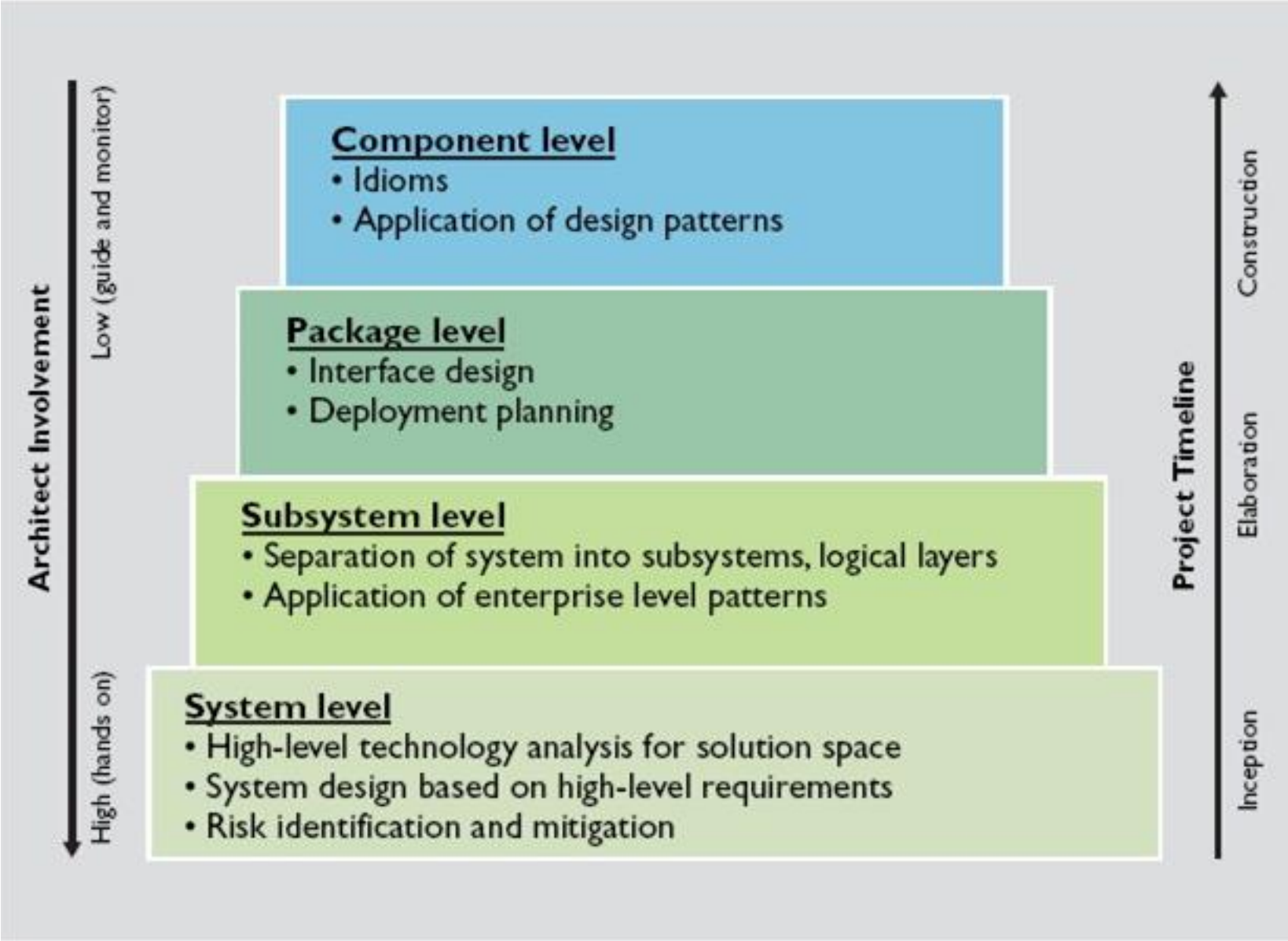
Why is architecture important?

- Martin Fowler in *Patterns of Enterprise Application Architecture:*

  "The highest-level breakdown of a system into its parts; the decisions that are hard to change; there are multiple architectures in a system; what is architecturally significant can change over a system's lifetime; and, in the end, architecture boils down to whatever the important stuff is."

- **Solution communication** and **consensus** among stakeholders
- **Earliest** and most **fundamental design** analysis and decisions
  - ***Directs and constrains remaining software development, deployment, and maintenance***
  - ***Dictates structure of development organization***
  - ***Enables early evolutionary prototyping***
  - ***Enables more accurate cost and schedule estimation***

Architecture is the highest-level design of a system



The Software Architect, by Matthew McBride (COMMUNICATIONS OF THE ACM May 2007/Vol. 50, No. 5)

Requirements affect the system architecture

- Non-functional requirements (NFRs) and constraints lead to a logical and physical architecture.
- Operational NFRs:
  - ***Scalability***
  - ***Availability …***
- Developmental NFRs:
  - ***Testability***
  - ***Portability …***
- Constraints:
  - ***Pre-chosen system components (eg, database)***
  - ***Pre-chosen frameworks …***

Architecture is guided by principles and patterns

- Manage risk
- Build vs buy vs open source
- Separation of concerns
- Architectural patterns are selected to satisfy NFRs
  - ***Failover*** and ***Load Balancing***
  - ***Model-View-Controller***
  - ***Tiers and Layers***
    - For example: UI, Application and Model
  - ***Java EE Patterns***
- No one architecture is right or wrong, just more or less useful for a given application. (attribution unknown)
  - ***Does it satisfy the NFRs?***
  - ***Ad-hoc architecture is not very useful.***

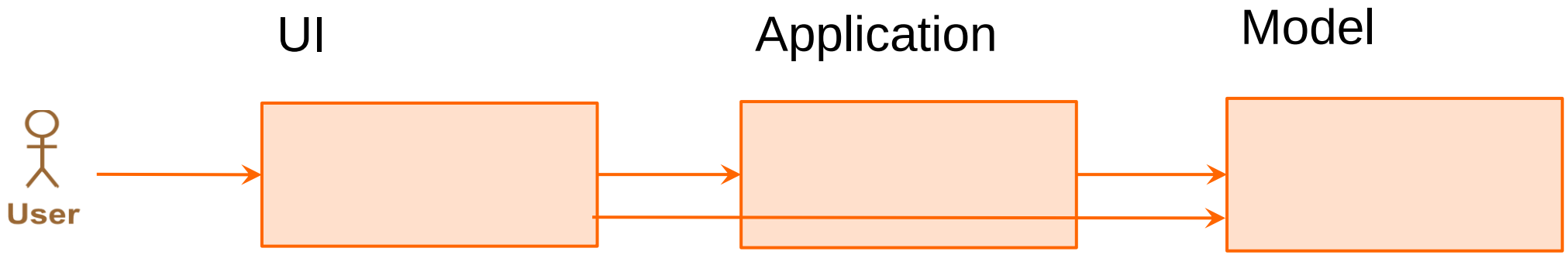
Visualize the architecture in the Inception phase

- The initial architecture is visualized during the Inception phase.
  - ***Recorded in the Vision document***
- It provides broad ideas:
  - ***Desktop app***
  - ***Web app***
  - ***Mobile app***
- Recommended frameworks are listed.
  - ***UI framework?***
  - ***Frontend/backend communication?***
  - ***Data storage?***

Build out the architecture in the Elaboration phase

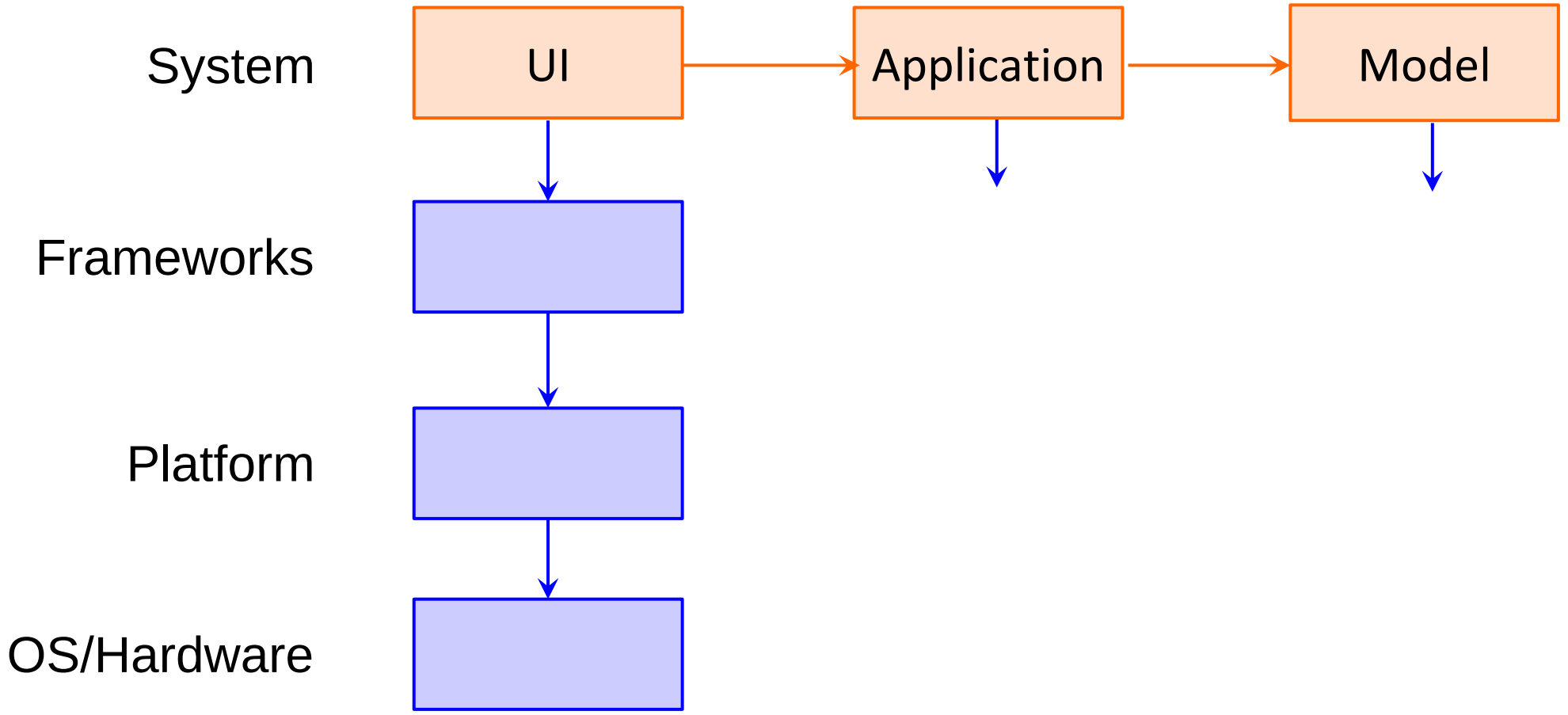
- The architecturally-significant user stories are prioritized during the Elaboration.
- The development team is frequently guided or lead by an architect during this phase.
- The working increment at the end of Elaboration forms the starting point of the system architecture.
  - ***There will be architectural additions over the lifetime of the system.***
  - ***Avoid changing established architectural norms.***

Architecture is modeled using tiers

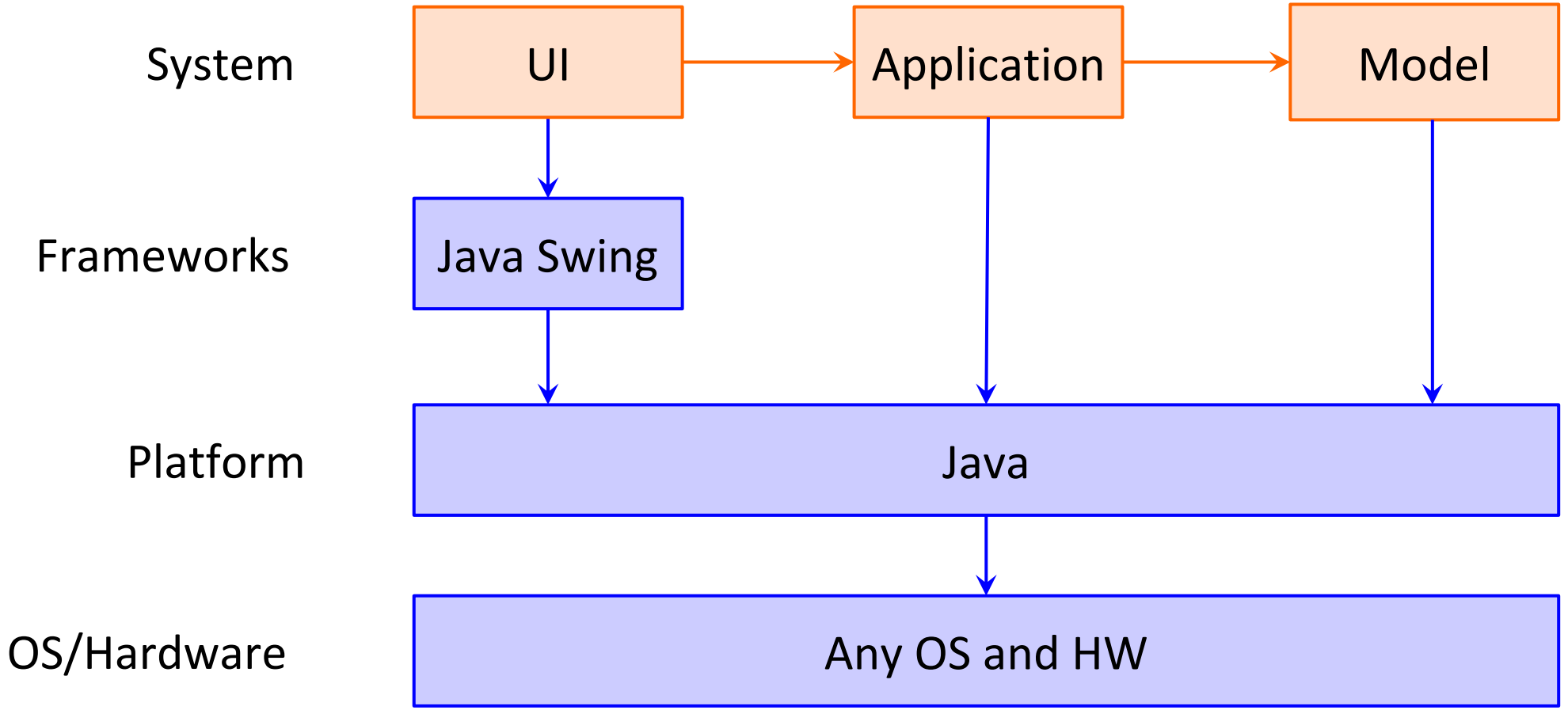


- The User interacts with the User Interface (UI) tier.
- The UI tier interacts with the Application and Model tiers.
- The Application tier holds logic that controls the flow of the application.
- The Model tier holds the core domain (aka "business") logic.

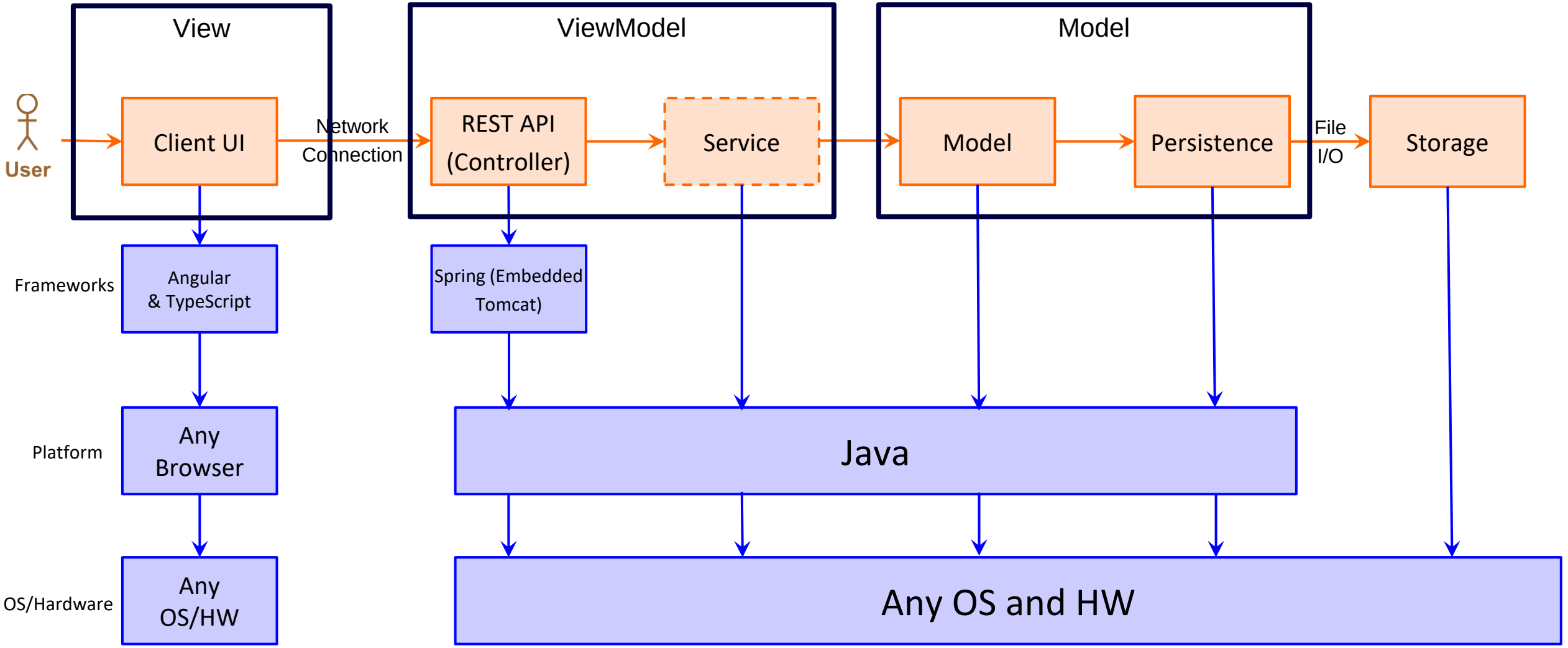
Architecture must also consider layers



Let's start with a simple, desktop architecture

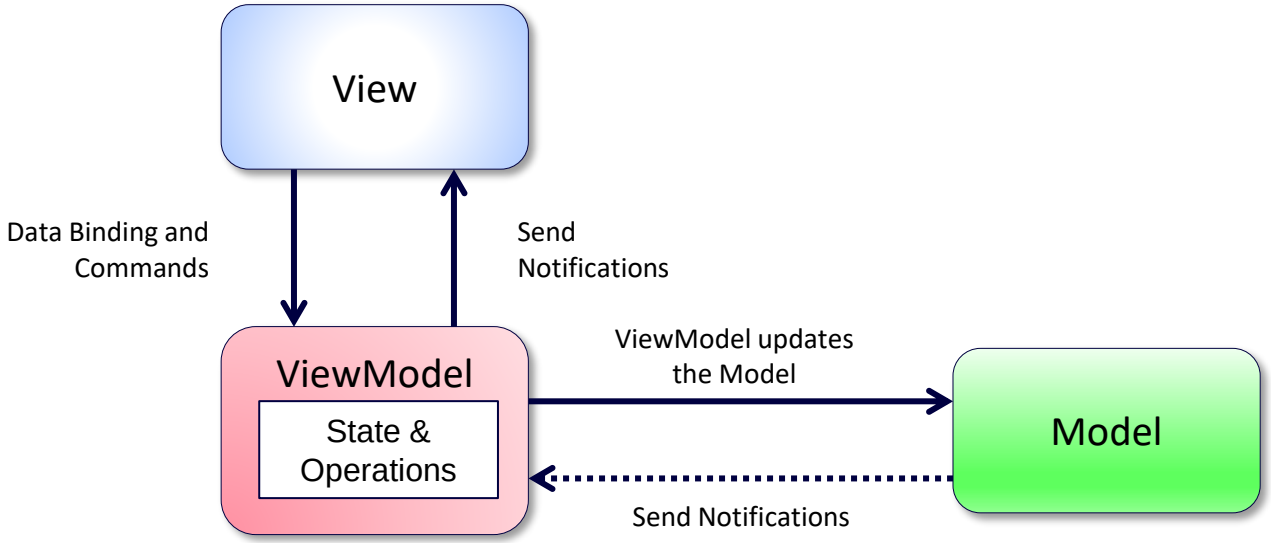


This is the architecture for an Angular app



Model-View-ViewModel (MVVM)

- The Model-View-ViewModel (MVVM) pattern helps to cleanly separate the business and presentation logic of an application from its user interface (UI).
  - ***Maintaining a clean separation between application logic and the UI helps to address numerous development issues and can make an application easier to test, maintain, and evolve.***
  - ***It can also greatly improve code re-use opportunities and allows developers and UI designers to more easily collaborate when developing their respective parts of an app.***



The view ***knows about*** the view model, and the view model ***knows about*** the model, but the model is ***unaware*** of the view model, and the view model is ***unaware*** of the view.

The view model isolates the view from the model classes to evolve independently of the view.