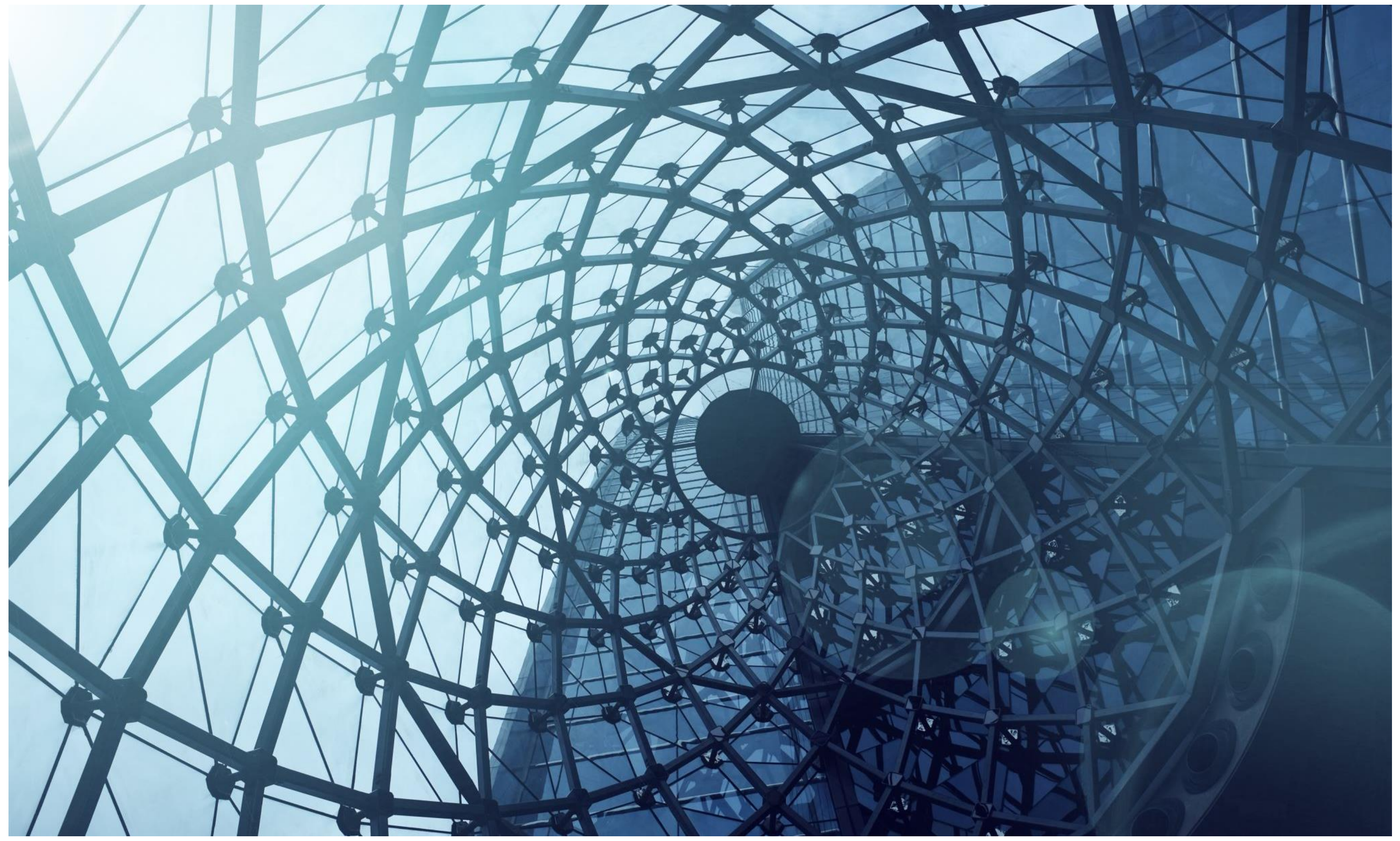


SOFTWARE ARCHITECTURE

Principles, Data Modeling, and Serialization

ANALOGIES FOR SOFTWARE DEVELOPMENT

Discuss

The Analogy of Urban Planning

- Lots of repetition
- Constrained variations
- Roads and Access
- Need similar materials
- Water and Sanitation
- Electricity

NEED TO BE LIVABLE

Before it is finished

NEEDS TO SUPPORT FUTURE CONSTRUCTION

Plan to extend

NEEDS TO ADJUST TO CHANGING NEEDS

Usage might not be what is expected.

A Balancing Act Between

YAGNI – You Aren't Going to Need It

Big Ball of Mud - aka Spaghetti Code

GOAL OF SOFTWARE ARCHITECTURE

Reduce cost and time to develop, modify, and extend software

In Concrete Terms

Once a module is completed can be reused → Code is easy to test and debug → Implementation is quick and easy

Architectural Principles

Separation of concerns
- Implies - persistence ignorance

Don't repeat yourself (DRY)
- No redundancy

Single responsibility
- Each module does one thing

Encapsulation
- Implies - explicit dependencies

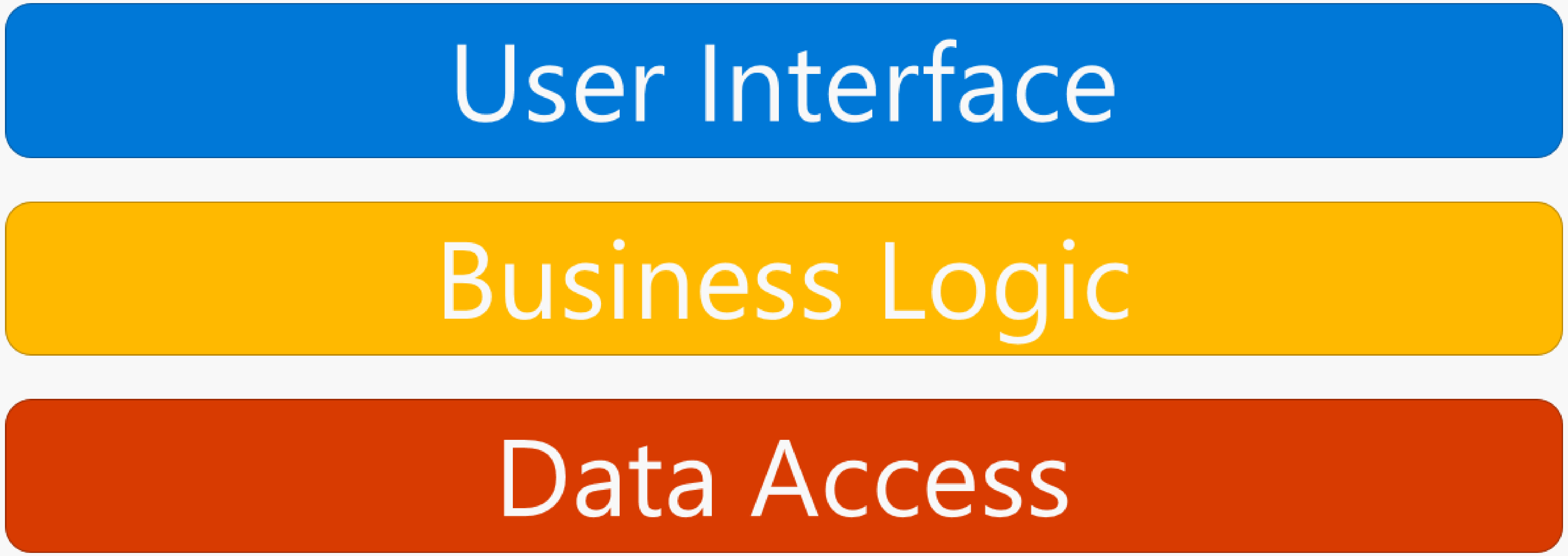
Separating UI from Business Logic

- The text-book example of separation of concerns and single responsibility
- Let's you concentrate on each aspect of the problem separately—and one complicated thing at a time is enough.
- It also lets different people work on the separate pieces, which is useful when people want to hone more specialized skills.
- See: https://www.martinfowler.com/ieeeSoftware/separation.pdf

PERSISTENCE IGNORANCE

How data is stored is independent of how it is used

Application Layers



https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures

Breaking up an SVG Editor Program

- How do we
  - Separate UI from Business Logic?
    - Can it be tested without UI?
    - Can a new UI library be used?
  - Make sure that operations can be easily
    - Logged
    - Repeated
    - Debugged

What is a Data Model

An abstract formalization of the objects and relationships found in a particular domain.

What are the primary domains?

Problem domain - the ideas and processes of the user

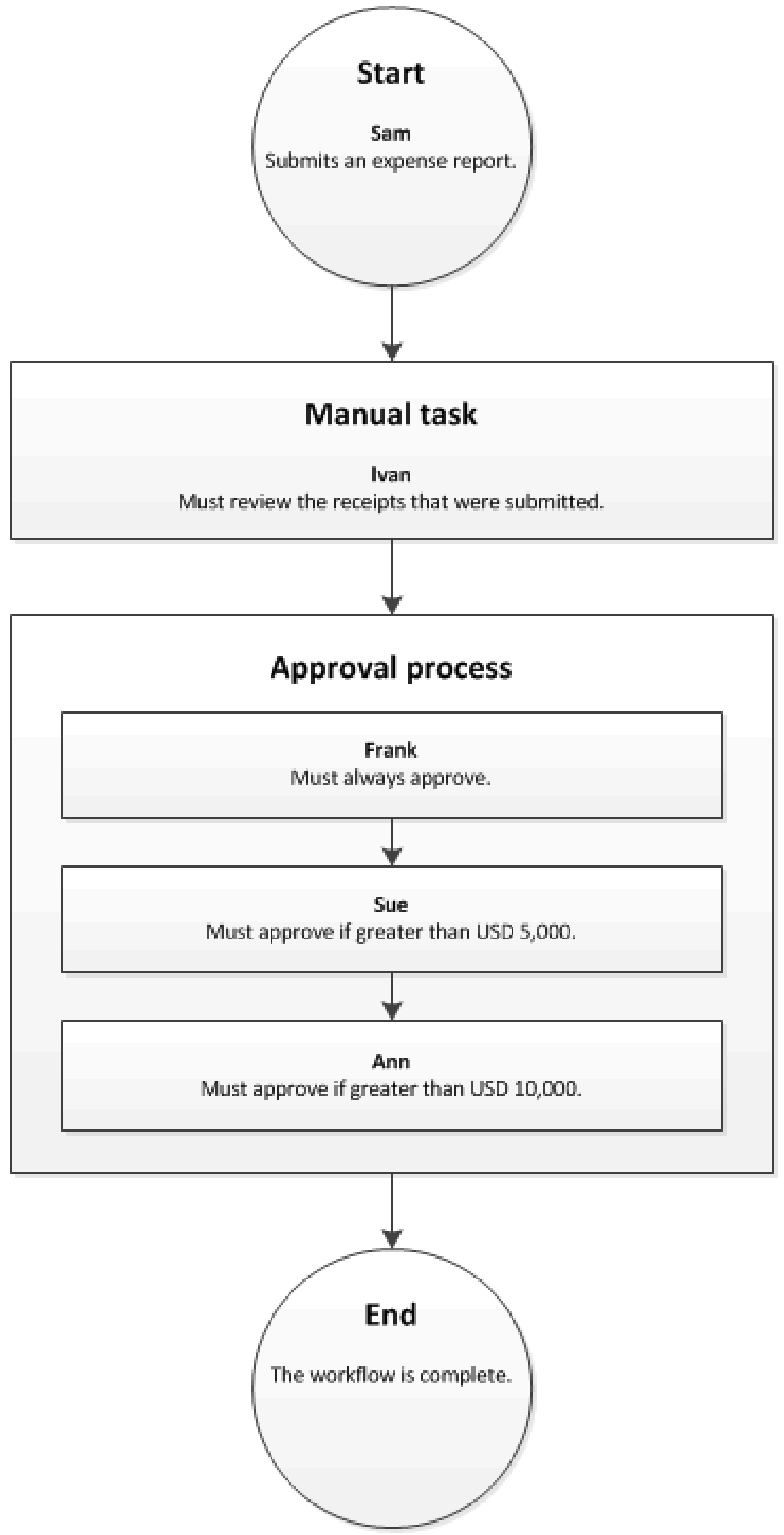
Solution domain – the ideas and processes of the implementation

Domain Driven Design (DDD)

In DDD structure and language of software code (class names, class methods, class variables) should match the business domain.

For example, if software processes loan applications, it might have classes like loan application, customer, and methods such as accept offer and withdraw.

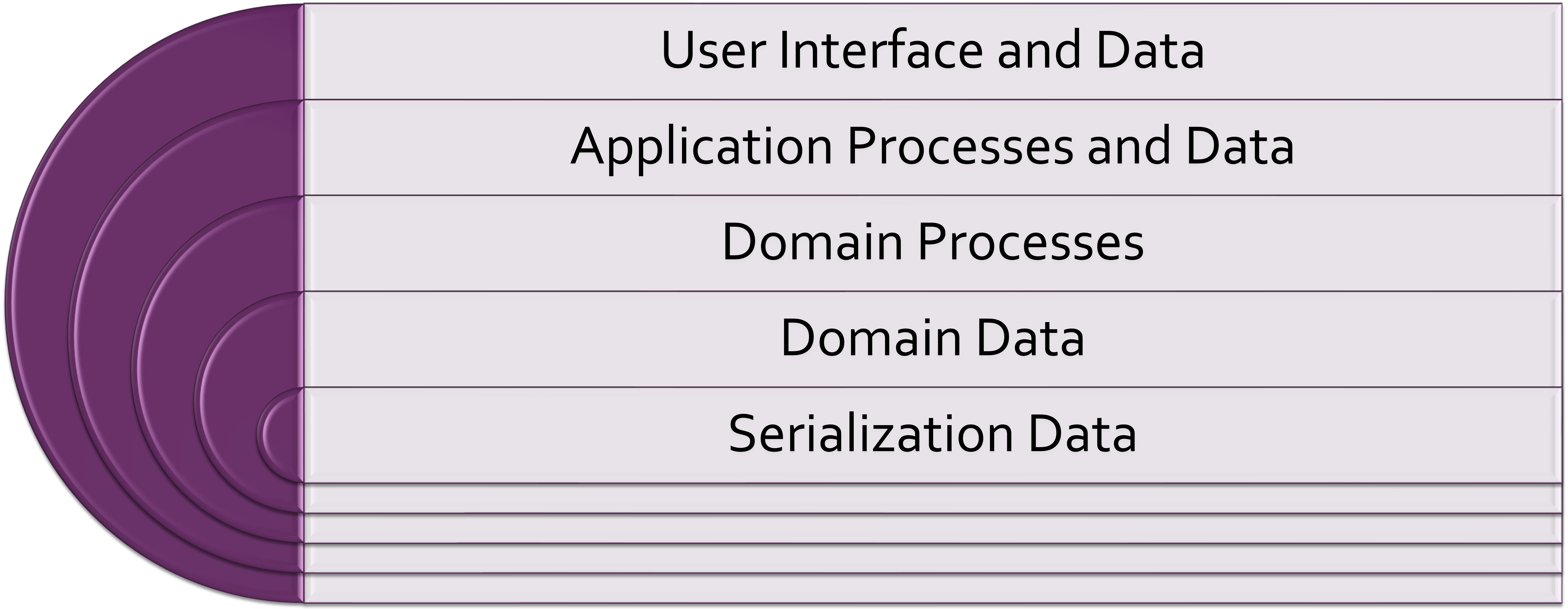
Very important and influential idea, related to Object Oriented Design (OOD)

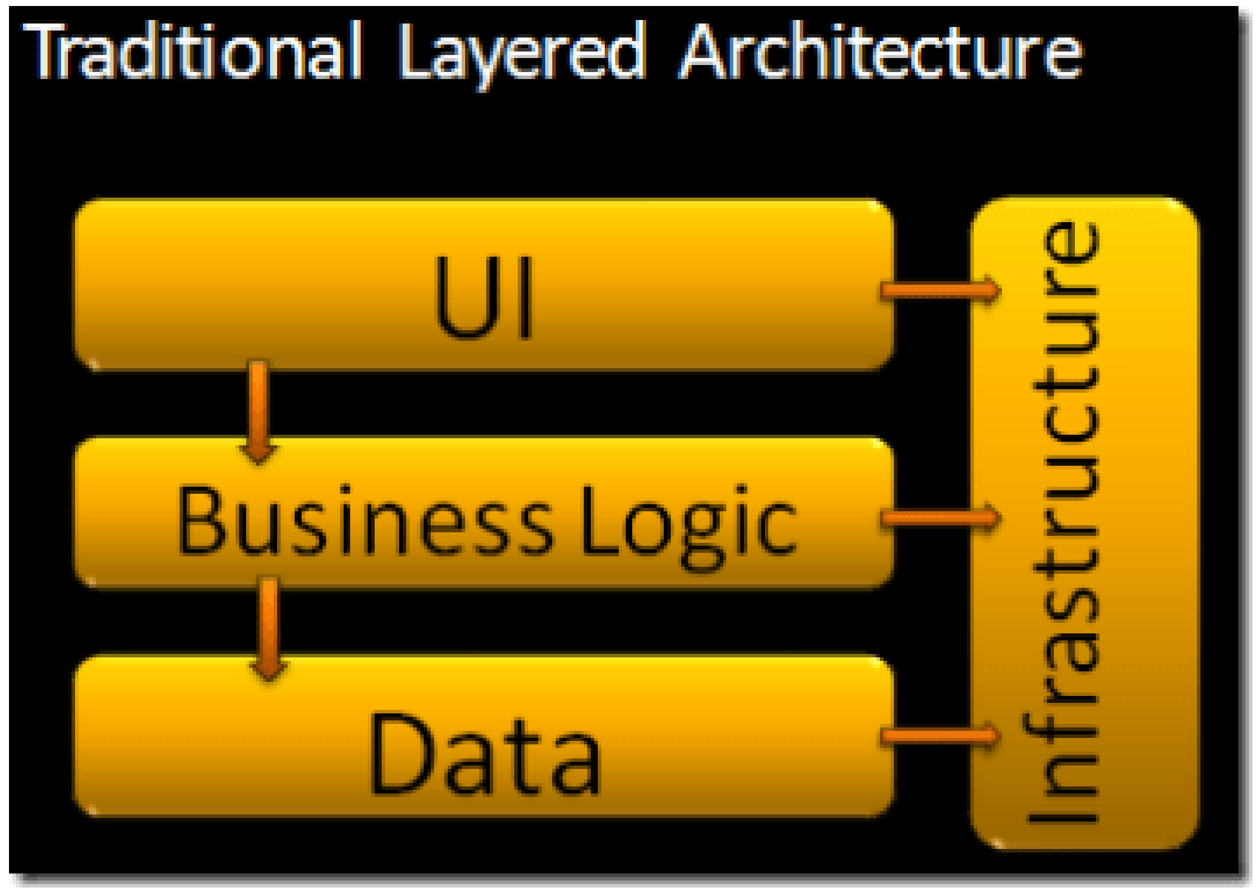

Start
Sam
Submits an expense report.
Manual task
Ivan
Must review the receipts that were submitted.
Approval process
Frank
Must always approve.
Sue
Must approve if greater than USD 5,000.
Ann
Must approve if greater than USD 10,000.
End
The workflow is complete.

MODELS CAN INCLUDE WORKFLOWS (PROCESSES)

https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/fin-ops/organization-administration/overview-workflow-system?context=%2Fdynamics365%2Fcontext%2Fcommerce

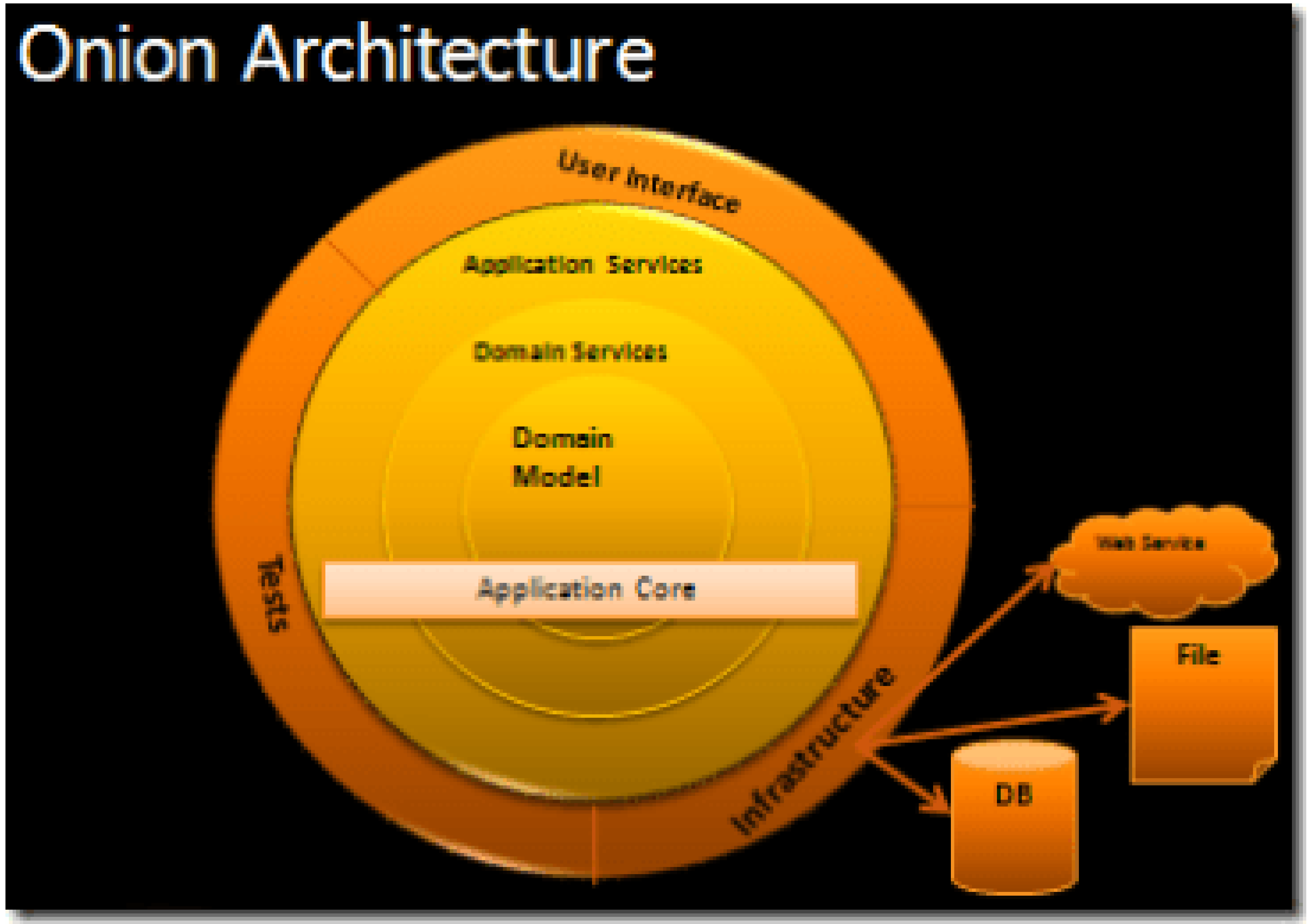
Software Architecture





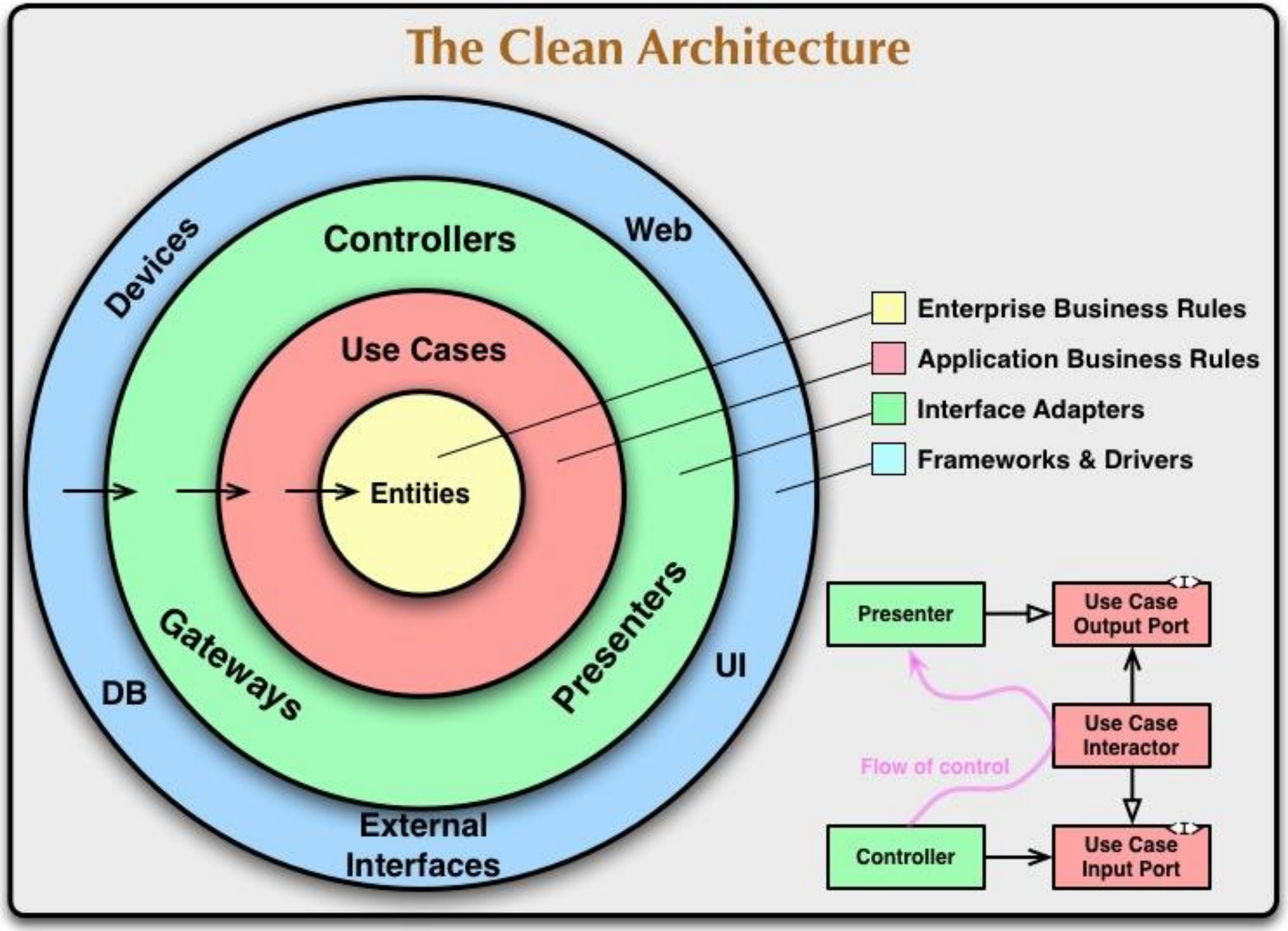
TRADITIONAL LAYERED ARCHITECTURE

https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/



https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/

ONION ARCHITECTURE



https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

How to Model Data

- Look for nouns in the problem domain
- Look for groupings
- Create a taxonomy
- Objects are great at that!

MODELING DOMAIN DATA OF AN SVG EDITOR

Discuss

Big Ball of Mud

- A BIG BALL OF MUD is haphazardly structured, sprawling, sloppy, duct-tape and bailing wire, spaghetti code jungle. We've all seen them. These systems show unmistakable signs of unregulated growth, and repeated, expedient repair. Information is shared promiscuously among distant elements of the system, often to the point where nearly all the important information becomes global or duplicated.

- https://blog.codinghorror.com/the-big-ball-of-mud-and-other-architectural-disasters/

Changing the Model

- In between the model and the view things are unclear
- What responsibilities do what?

The Goal of GUI Architectural Patterns

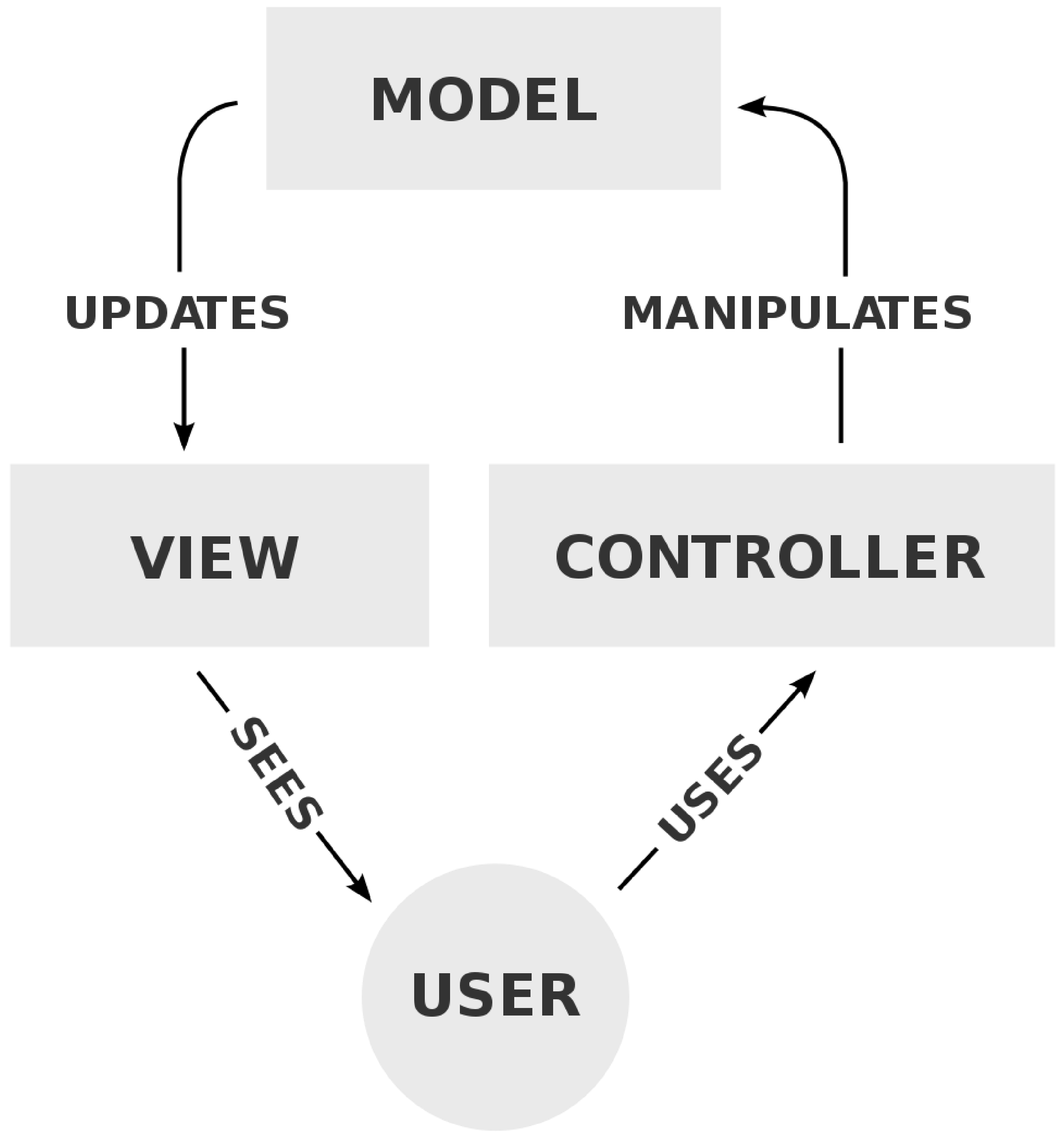
- Make GUI applications more flexible, and reusable.
- Can I port the application to a new platform?
- Can I change the UI without changing the rest of the code?
- Can I run the application without a UI?

Models, View, Controller (MVC)

Model is the state of the application and the rules of its behavior

View is the visual representation of data

Controller communicates user interactions to the model



MVC

- The user sees the view
- Interactions are sent to the controller
- The controller updates the model

MVC in ASP Web Applications

Models:

- Classes that represent the data of the app and use validation logic to enforce business rules. Model objects might retrieve and store model state to/from a database.

Views:

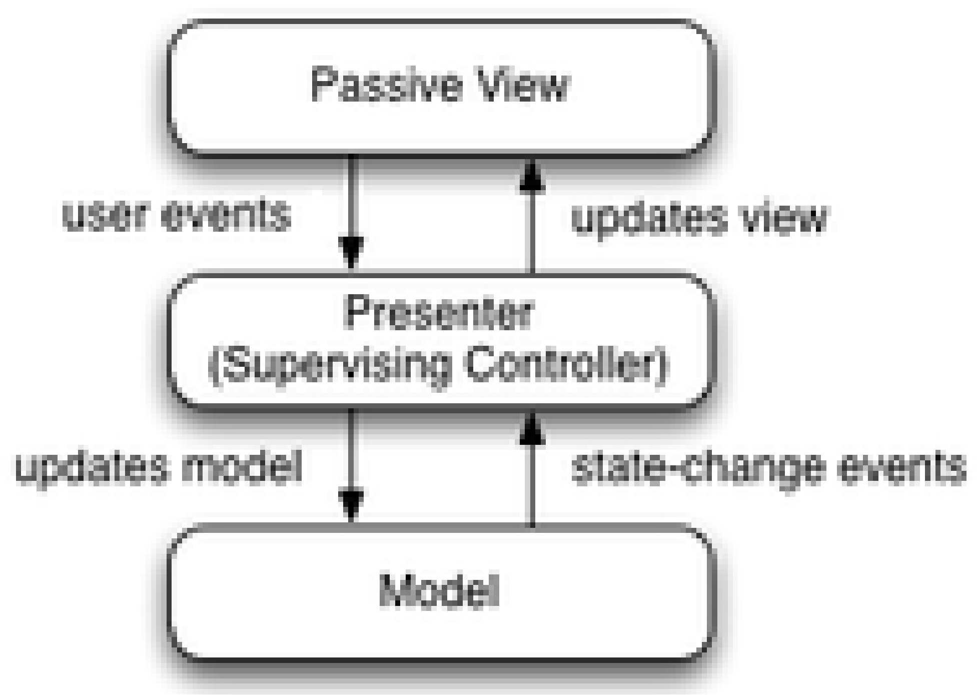
- Components that display the app's user interface (UI). Generally, this UI displays the model data.

Controllers:

- Classes that handle browser requests, retrieve model data, call view templates that return a response.

https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-controller

Presentation Model



- Back in 2004, Martin Fowler published an article about a pattern named Presentation Model (PM).
- The PM pattern is similar to MVP in that it separates a view from its behavior and state. The interesting part of the PM pattern is that an abstraction of a view is created, called the Presentation Model.
- A view, then, becomes merely a rendering of a Presentation Model.

MVVM – Model, View, View-Model

- Unlike the Presenter in MVP, a ViewModel does not need a reference to a view.
- The view binds to properties on a ViewModel, which exposes data contained in model objects and other state specific to the view.
- If property values in the ViewModel change, those new values automatically propagate via data binding.
- When the user clicks a button in the View, a command on the ViewModel performs the requested action.
- The ViewModel, never the View, performs all modifications made to the model data.
- https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern

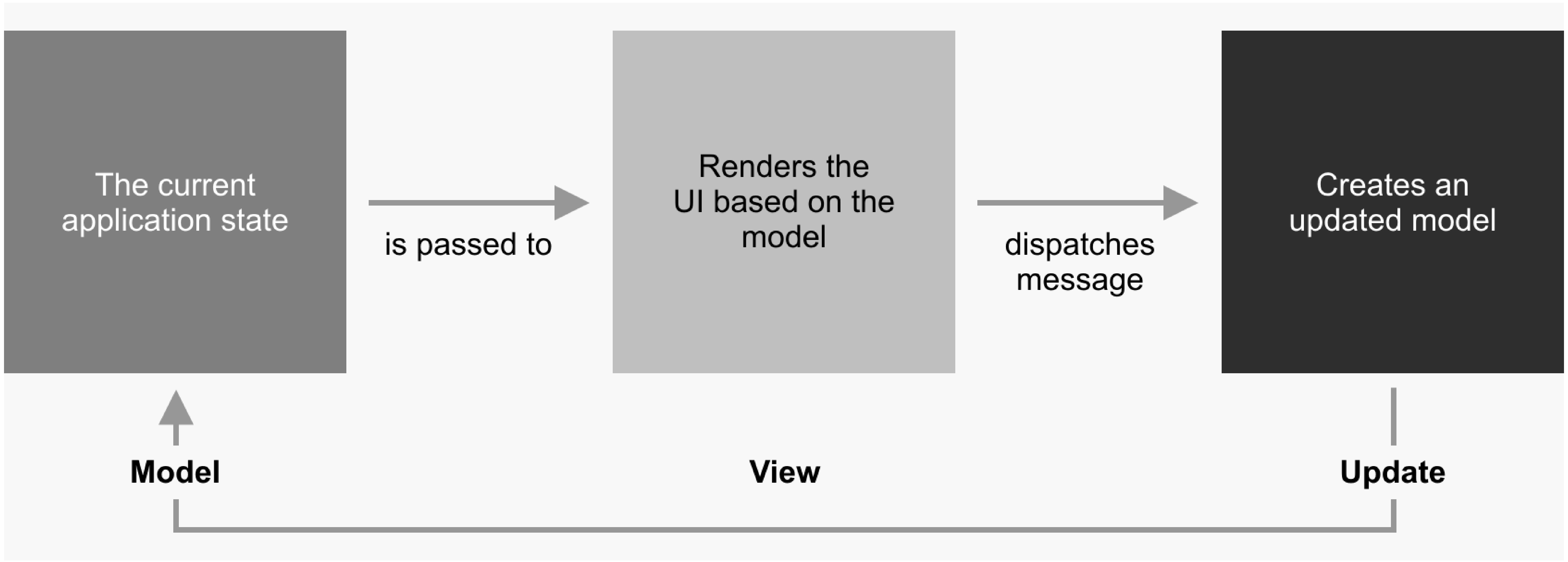
Model View Update (MVU)

A very simple idea

Model is changed through a formal "update" process

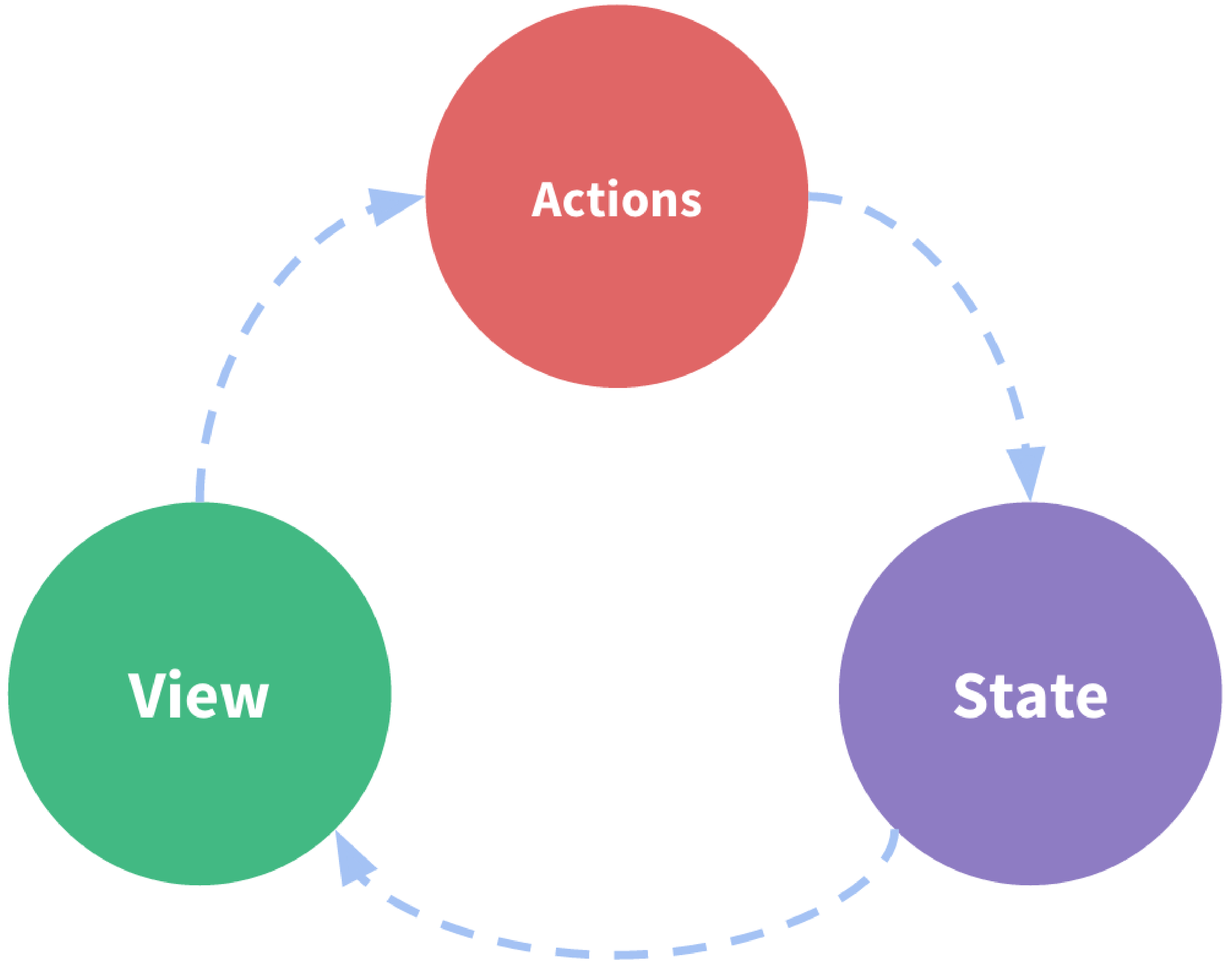
When the model changes the view is redrawn

Also called the "Elm Architecture"



https://thomasbandt.com/model-view-update

Redux Pattern



https://redux.js.org/tutorials/essentials/part-1-overview-concepts

IN A WINDOWS FORMS APPLICATION?

The form is both the view and the controller.