Software Engineering

With or Without You (U2)

Ivo Vondrak, 2023

- What is an engineering?

- What is an engineering?
- Engineering is about making ideas alive!

Software Engineering

The most accepted definition

- Software engineering is the branch of computer science that deals with the **design, development, testing, and maintenance of software** applications. Software engineers apply **engineering principles** and knowledge of programming languages to build software solutions for end users.
- Engineering principles mean **systematic and disciplined approach**

Is there really a professional programmer who writes code without planning?

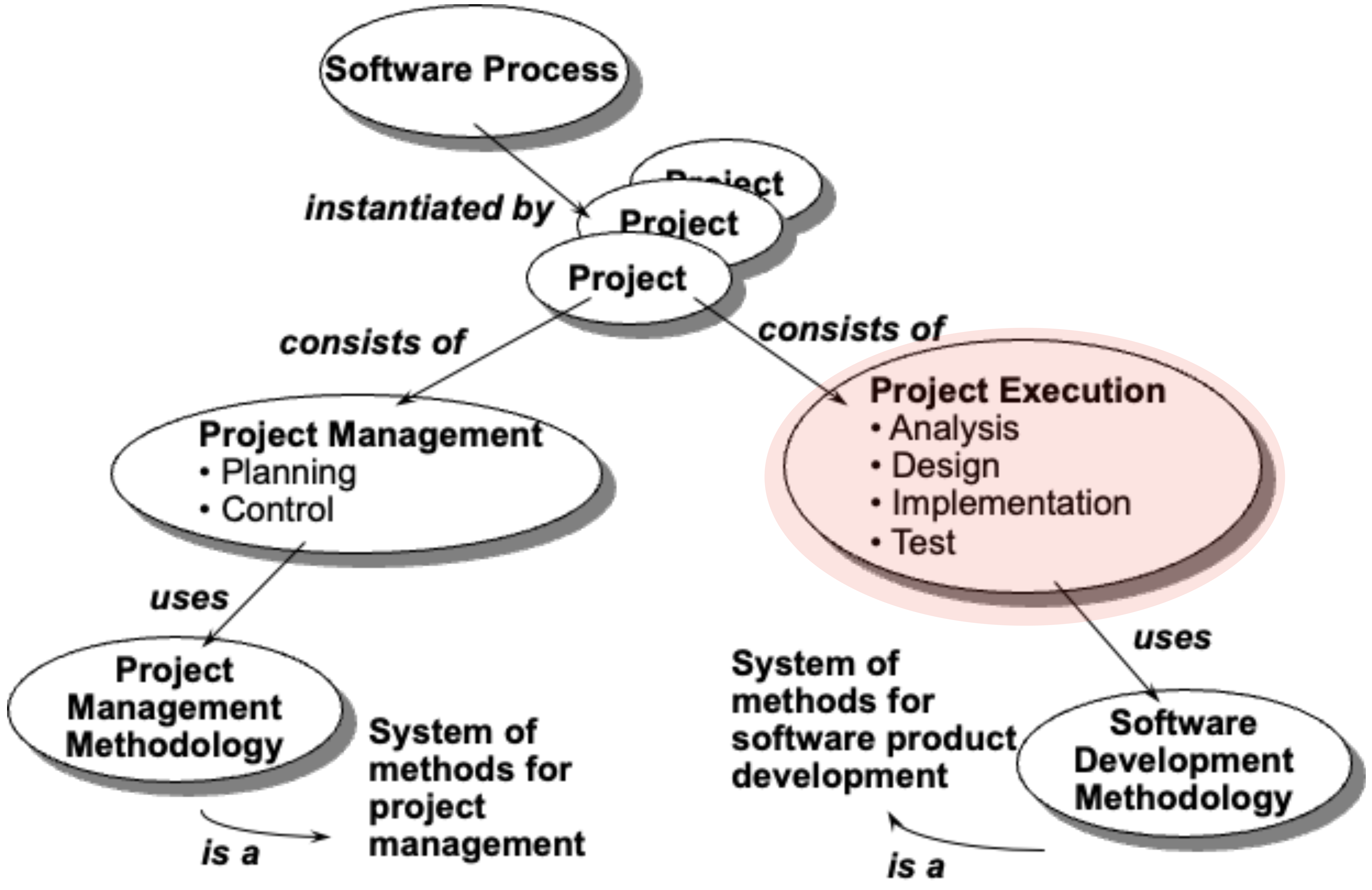
A physician a civil engineer, and a computer scientist were arguing about what was the oldest profession in the world. The physician remarked, "Well, in the Bible, it says that God created Eve from rib taken out of Adam. This clearly required surgery, and so I can rightly claim that mine is the oldest profession in the world. " The civil engineer interrupt, and said, "But even earlier in the book of Genesis, it states that God created the order of heavens and earth from out of chaos. This was the first and certainly the most spectacular application of civil engineering. Therefore, fair doctor, you are wrong: mine is the oldest profession in the world." The computer scientist leaned back in the chair, smiled, and then said“ confidently, **"Ah, but who do you think created the chaos?"**

Unknown but very experienced author 🤪

How to live with …

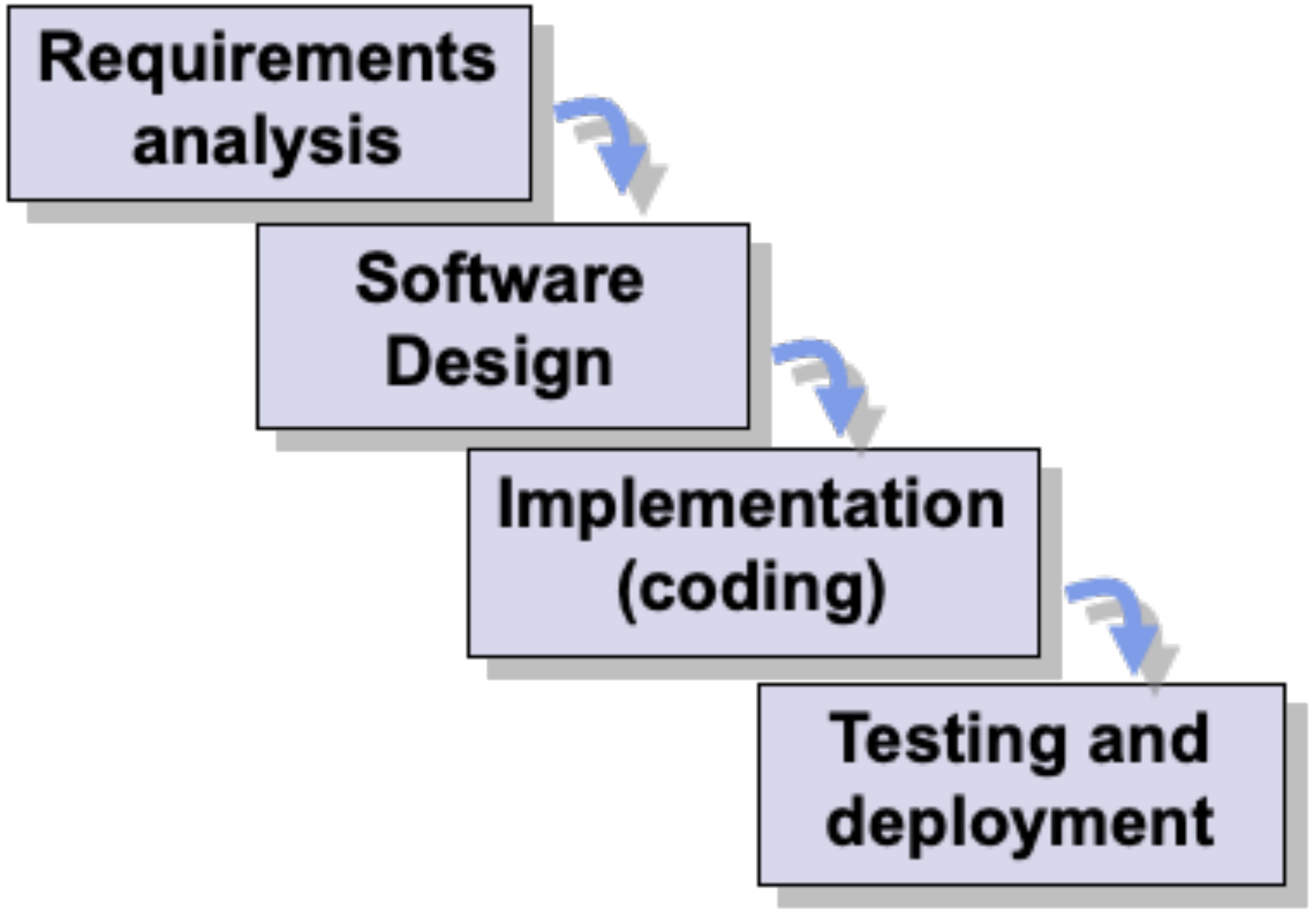
Software Production Layout

Think about it OR just code it 🤷‍♂️



The Water Fall Process Model

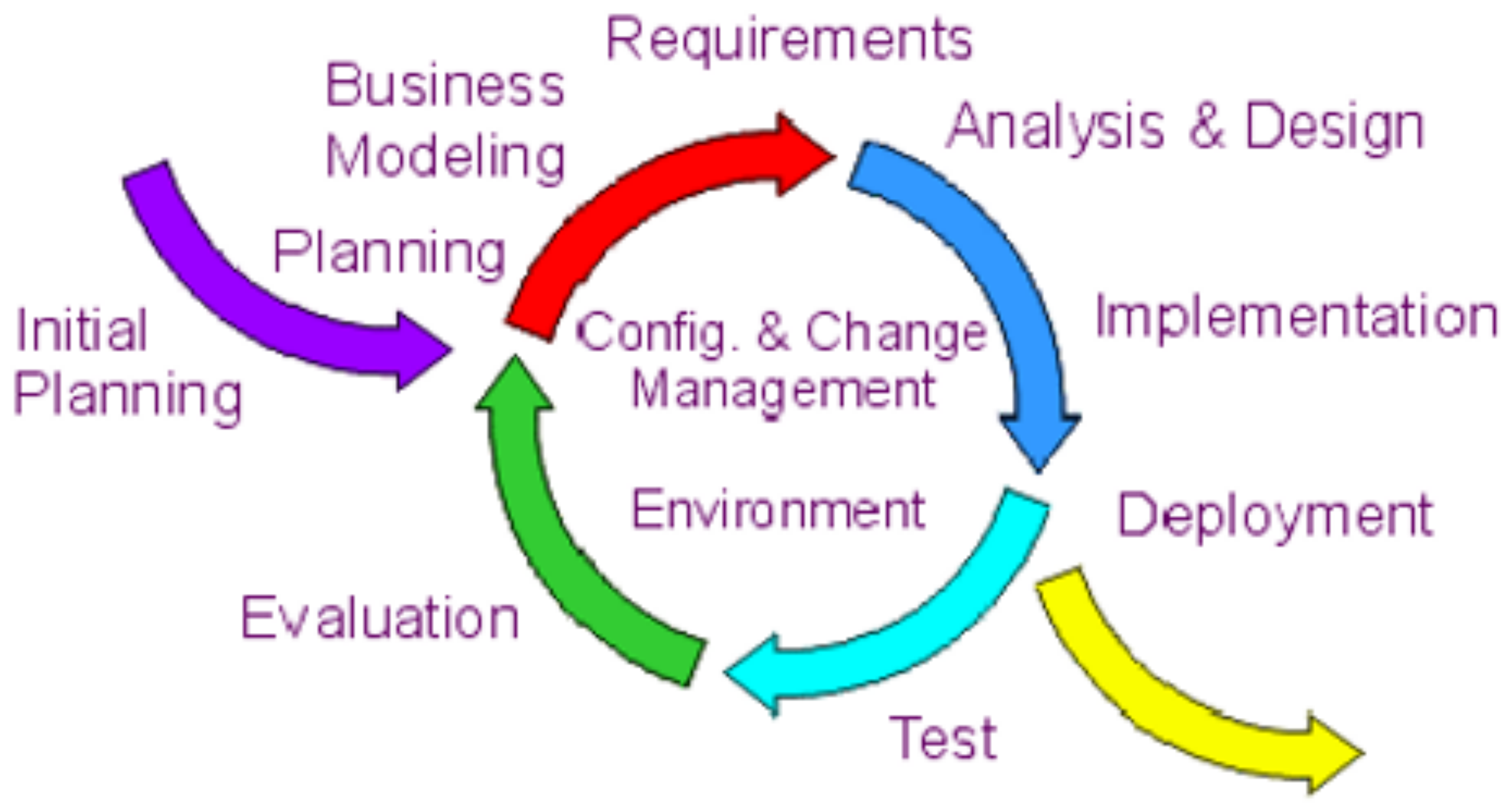
We are not building houses and bridges 🤷‍♂️



- It takes too long to see results.
- It depends on stable, correct requirements.
- It delays the detection of errors until the end.
- It does not promote software reuse and prototyping.

The Iterative Approach

Each phase of the software development can be further broken down into iterations. An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows incrementally from iteration to iteration to become the final system.🥇



Typical Iteration Flow

Benefits of an Iterative Approach

We do not know everything at the beginning 🙈

- **Risk Mitigation** – an iterative process lets developers mitigate risks earlier than a sequential process where the final integration is the only time that risks are discovered or addressed.
- **Accommodating Changes** – an iterative process lets developers take into account requirements, tactical and technological changes continuously.
- **Learning as You Go** – an advantage of the iterative process is that developers can learn along the way, and the various competencies and specialties are employed during the entire lifecycle.
- **Increased Opportunity for Reuse** – an iterative process facilitates reuse of project elements because it is easy to identify common parts as they are partially design and implemented instead of identifying all commonality in the beginning.
- **Better Overall Quality** – the system has been tested several times, improving the quality of testing. The requirements have been refined and are related more closely to the user real needs. At the time of delivery, the system has been running longer.

Architecture-Centric Development

Abstraction is the key player ☝️ ~~✌️~~

- Models help developers understand and shape both the problem and the solution.
- Model is a simplification of reality that help us master a large, complex system that cannot be comprehended easily in its entirety. The model is not the reality, but the best models are the ones that stick very close to reality.
- Multiple models are needed to address different aspects of the reality. These models must be coordinated to ensure that they are consistent and not too reduntant.

Architecture

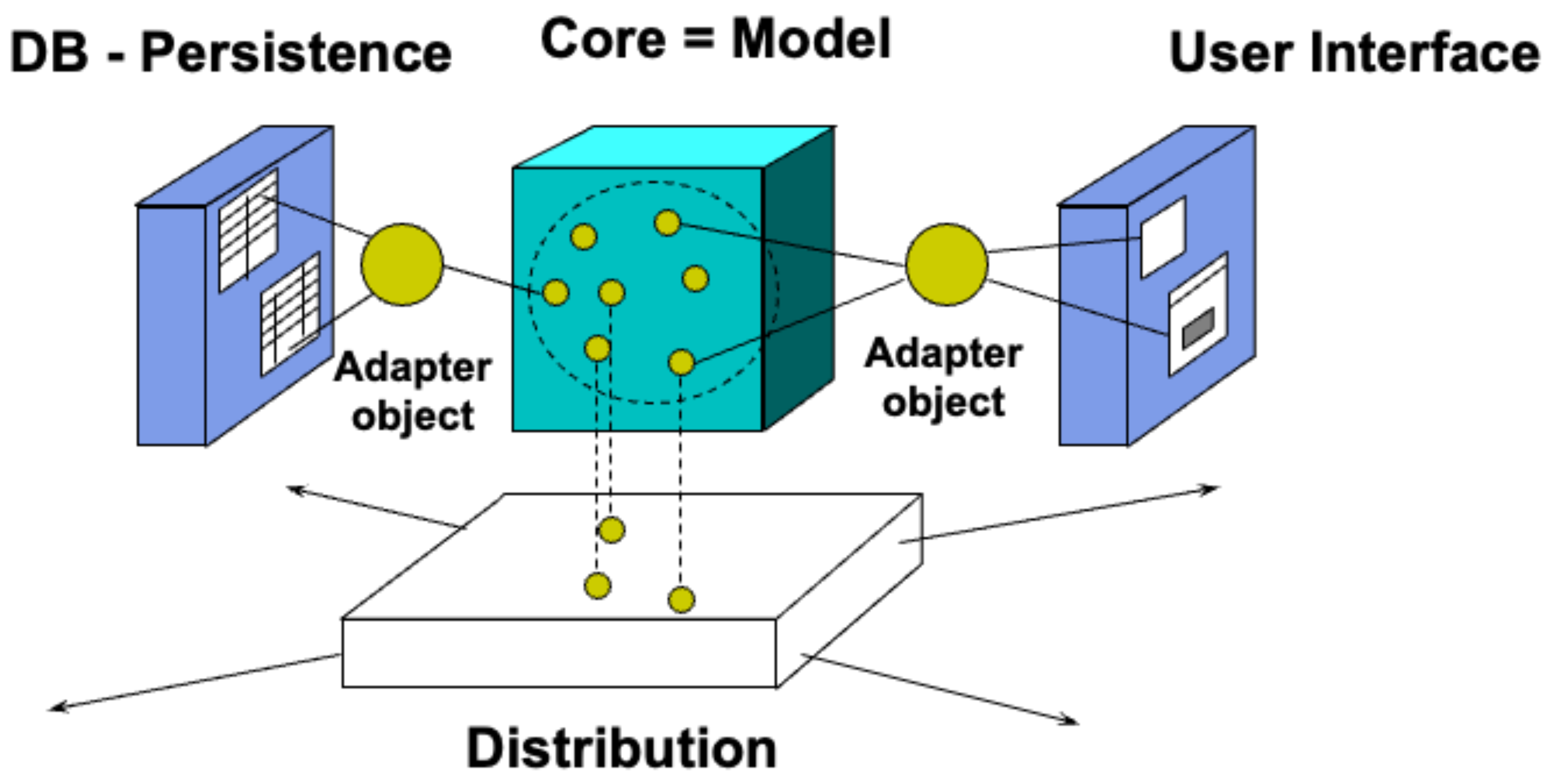
Architecture is the skeleton 🫥

- Architecture is what remains when you cannot take away any more things and still understand the system and explain how it works.

- Architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment

System Architecture

Fundamental organization of any software system



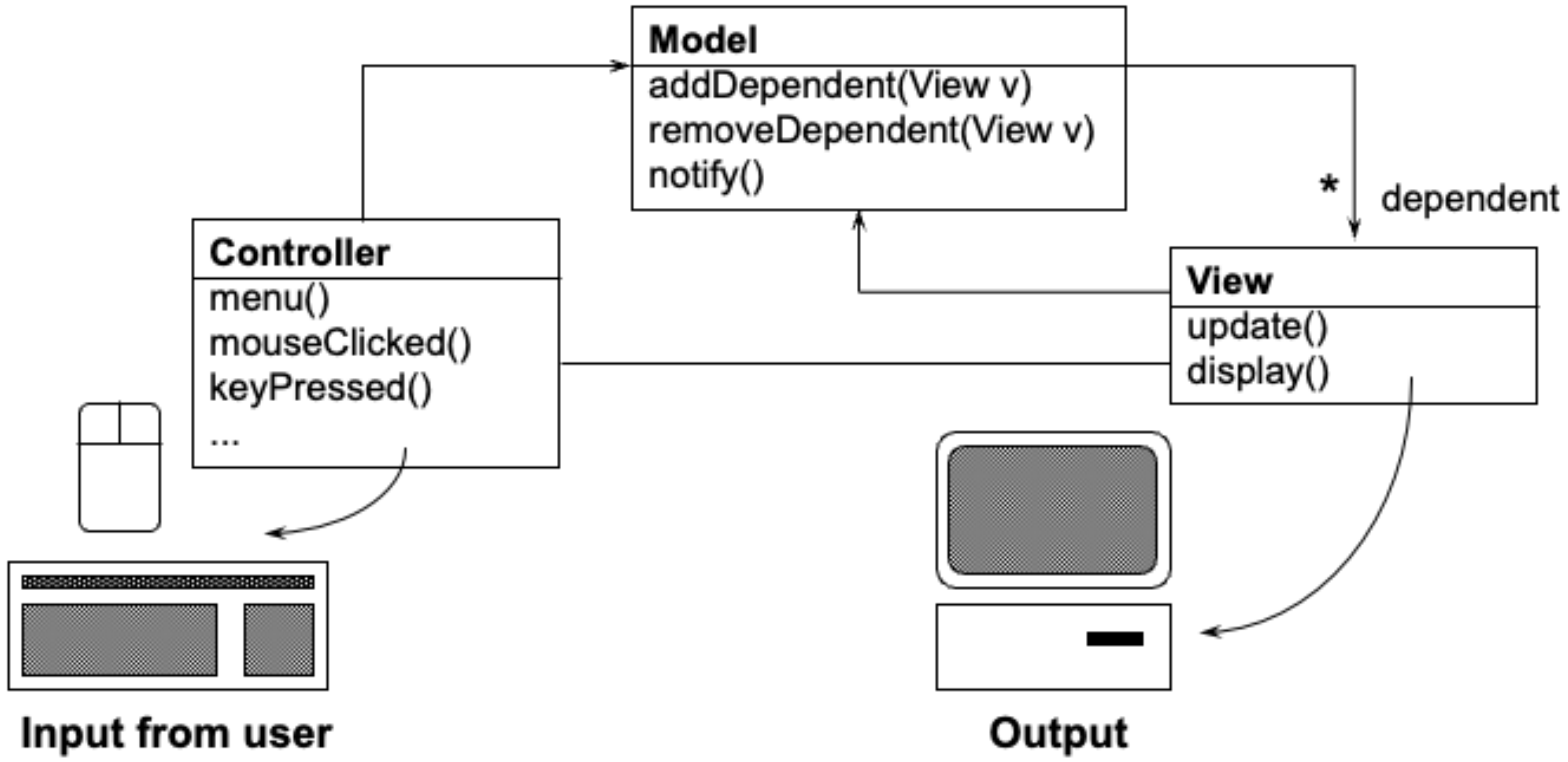
MVC – Application Framework

A Framework is a set of abstract and concrete classes that comprise a generic software system. Framework is calling our classes = full control of flow.

- Any application can be divided into **two parts**: (1) **The domain model**, which handles data storage and processing. (2) **The user interface**, which handles input and output.
- **Model-View-Controller** Architecture:
- **Model**: domain state and behavior (dynamic object).
- **View**: display current state of dynamic object.
- **Controller**: effect user-evoked behavior from the dynamic object.

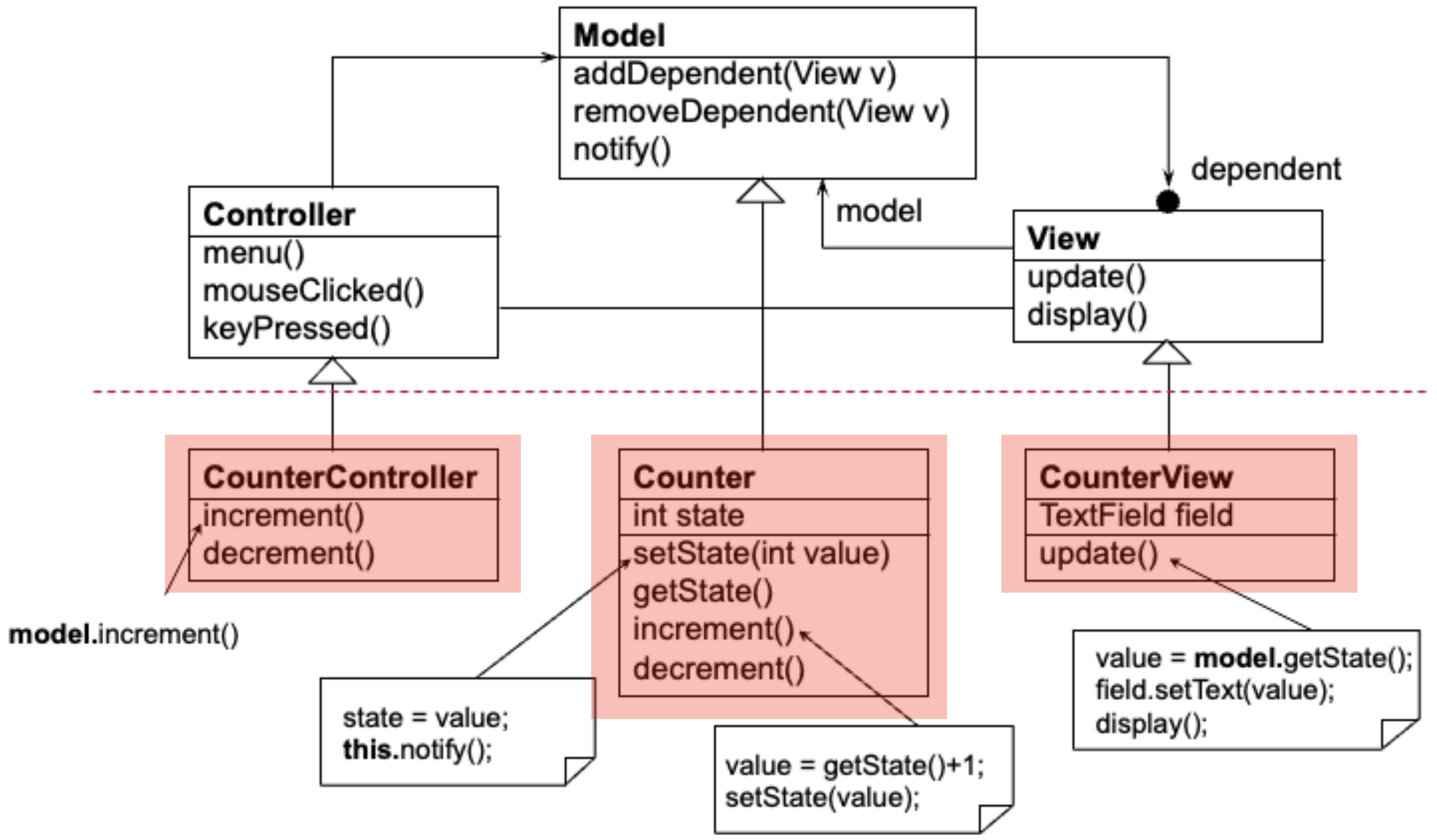
MVC Classes

Trygve Reenskaug created MVC while working on Smalltalk-79 as a visiting scientist at the Xerox Palo Alto Research Center (PARC) in the late 1970s.💡



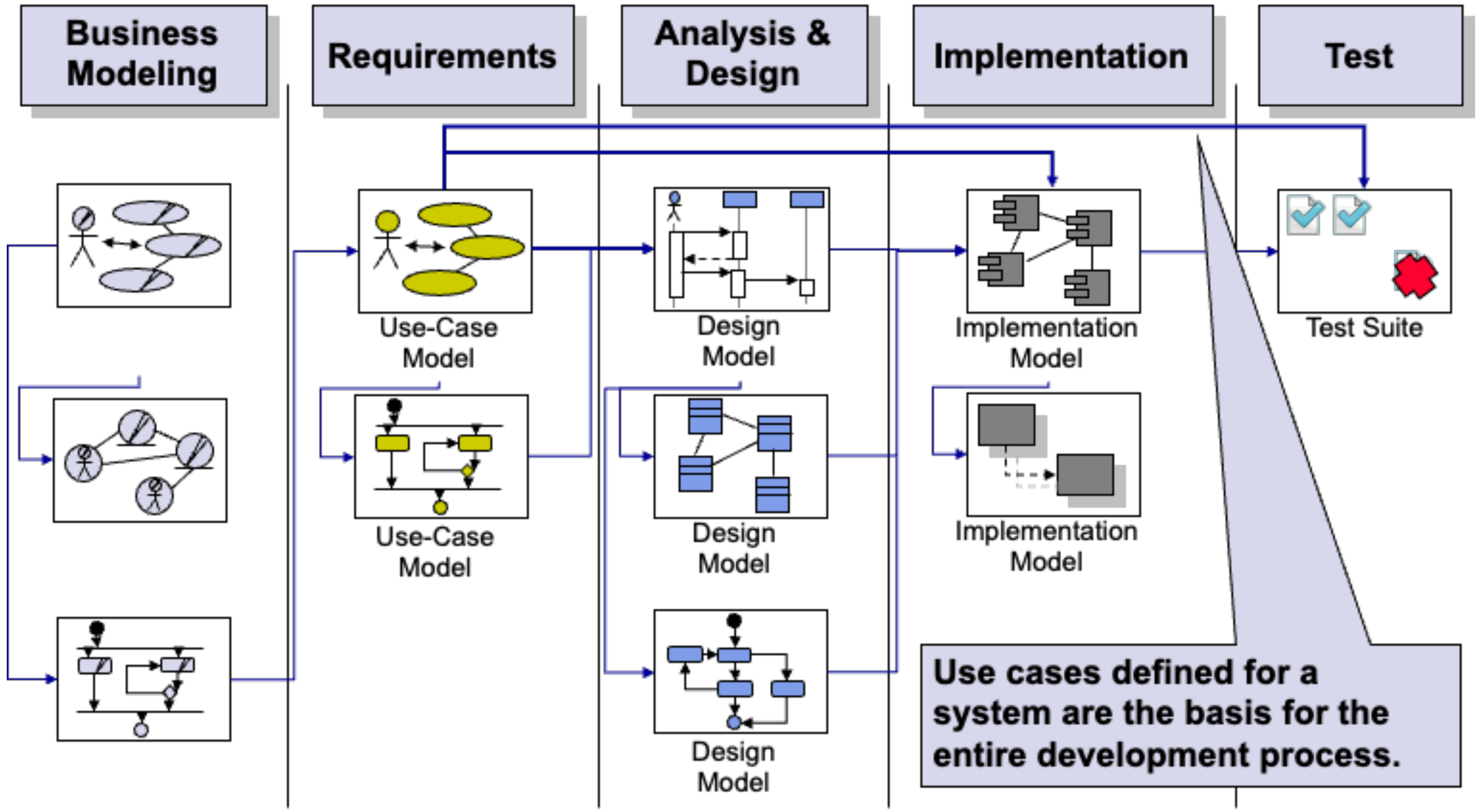
MVC Example

How to increment a decrement a counter.



Models - Blueprint of the Software

UML - Unified Modelling Language



… or without software engineering

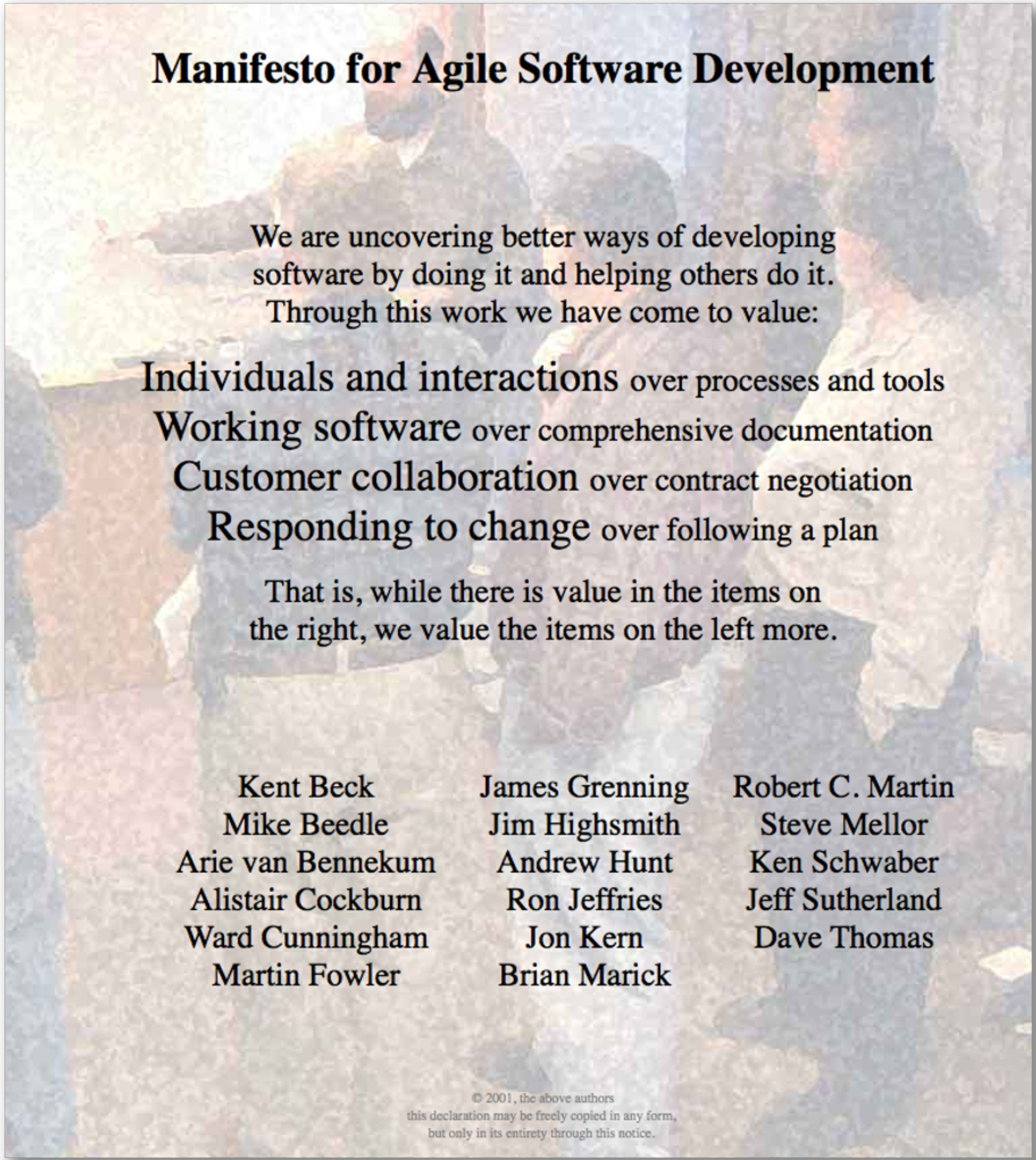
Other Approaches

Iterations are always included! ☝️

- **Spiral model** (Barry Boehm) – **evolutionary model** where each cycle produces something to be evaluated, but not it need not be a usable system.
- **Prototyping Model** - **simplified version** of the proposed system is presented to the customer for consideration as part of the development process.
- **Agile Software Processes** - agile methods attempt to minimize risk by **developing software in short timeboxes,** called iterations, which typically last one to four weeks. **Intensive communication between the developers and customers is assumed.**
- **eXtreme Programming** (Kent Beck) - XP focuses on frequent testing, integration, and user review. **It utilizes the actual source code as the design document** and "user stories" as requirements documents

I don't believe in coding design tools. I've been programming for more than 30 years now (40 years if you go back to my first class in programming). I think in code. Code is my design language and procrastination is my friend.

Peter Vogel 12/27/2016



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

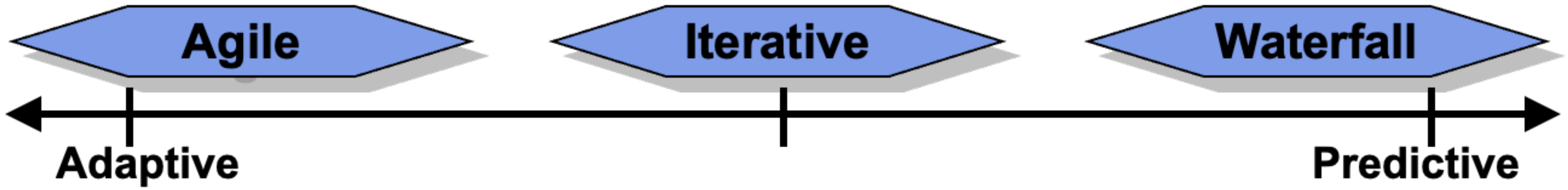
Summary

Adaptive vs. Predictive Methods

All approaches have their own pros and cons 🤷‍♂️

- **Adaptive methods** focus on **adapting quickly to changing realities**. When the needs of a project change, an adaptive team changes as well. An adaptive team will have difficulty describing exactly what will happen in the future.

- **Predictive methods focus on planning the future in detail.** A predictive team can report exactly what features and tasks are planned for the entire length of the development process. Predictive teams have difficulty changing direction. The plan is typically optimized for the original destination and changing direction can cause completed work to be thrown away and done over differently.


Agile
Iterative
Waterfall
Adaptive
Predictive