Software Engineering

How To Create Software Products Everyone Loves!

Module 1

Process Models

Ali Samanipour

Shiraz University of Technology
September. 2025

Choose the Right Tool for the Job

Keep an open mind when selecting a process to fit your project. It is a mistake to think that iterative or parallel software process models are always the best tools for the job. **Depending on the situation, a different process could be more effective**. Also, aspects of different processes can be combined to better suit the needs of the project.

Session Roadmap

1. Linear Models
2. Iterative Models
3. Parallel Models
4. Prototypes

Linear Models

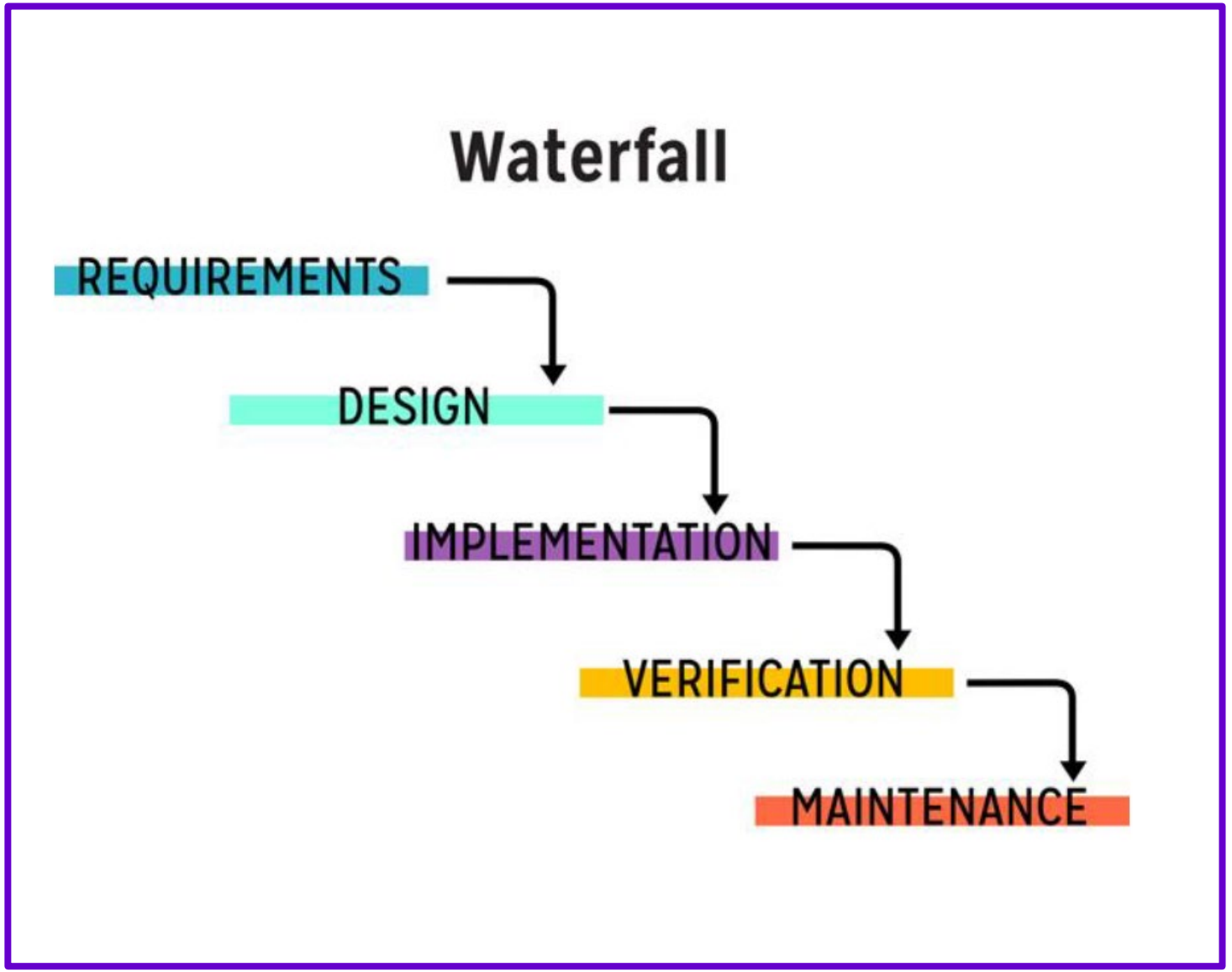
Linear process models follow a series of phases, one by one, with no return to prior phases

Linear processes are best suited to projects where very little feedback or refinement is expected.

Successful projects following a linear process model are well understood from the start, with little to no room for change.

Linear Models: Waterfall Model

The **Waterfall** model is a linear process model, in which each phase produces an approved work product that is then used by the next phase.



Waterfall Model: Benefits and Drawbacks

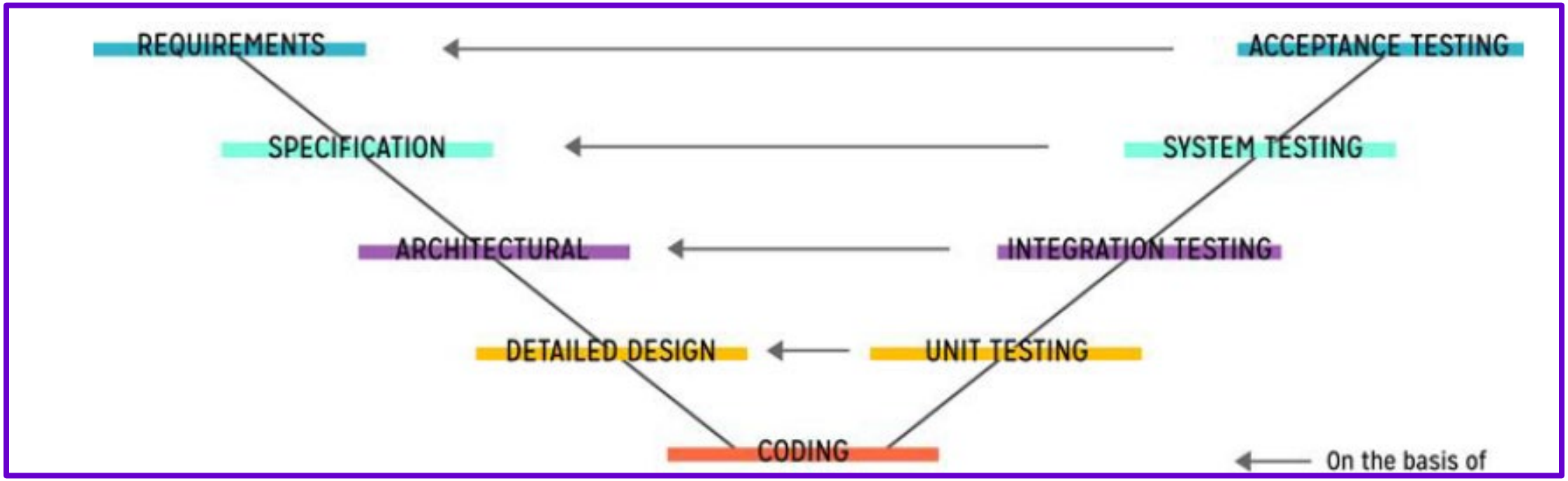
The Waterfall model focuses on knowing all the requirements up front. It is simply not designed to address mid-stream changes easily or without great cost, because it would require revisiting earlier phases.

The Waterfall model faces one major flaw: It is not very adaptable to change

Waterfall model does have benefits: easy to understand; clearly defines deliverables and milestones; emphasizes the importance of analysis before design, and design before implementation

Linear Models: V-Model

It was created in response to what was identified as a problem of the Waterfall model; that is, **it adds a more complete focus on testing the product**.



V-Model: Benefits and Drawbacks

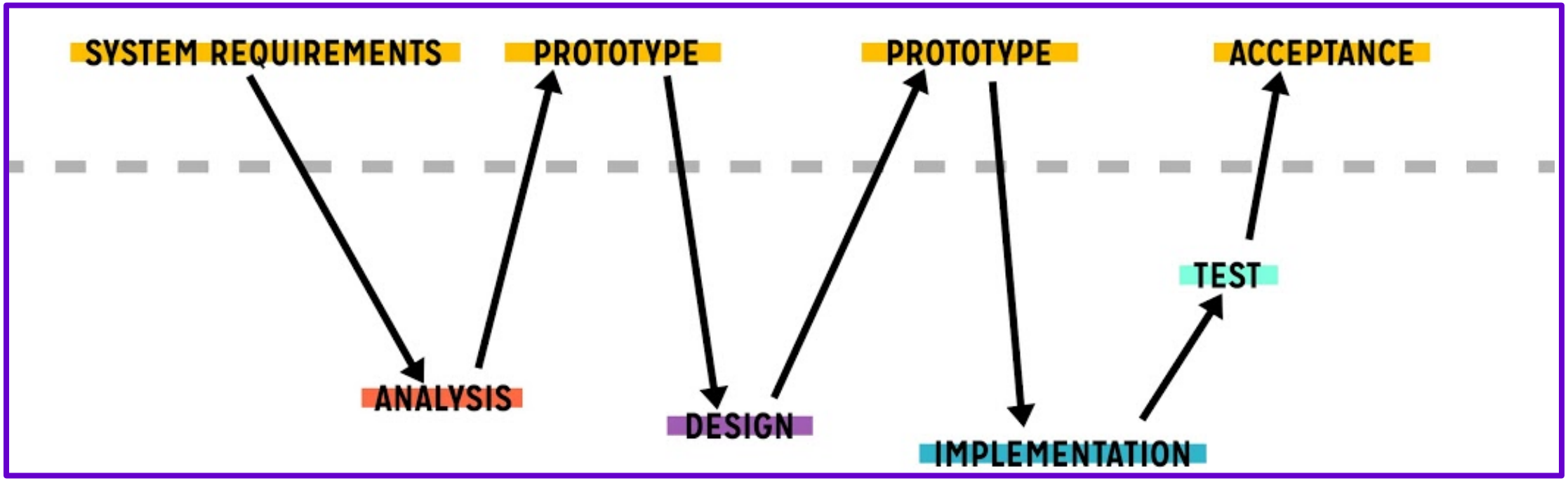
As the Waterfall model in that it is straightforward to understand, but it does not accommodate change well.

Major flaw: The client does not see the finished product until everything is virtually complete

Easy to understand; allowing the development team to verify the product at multiple levels in explicit phases is an improvement over the Waterfall model.

Linear Models: Sawtooth Model

As the process proceeds linearly, there are phases that involve the client in the top section of the diagram and phases that involve only the development team in the bottom section



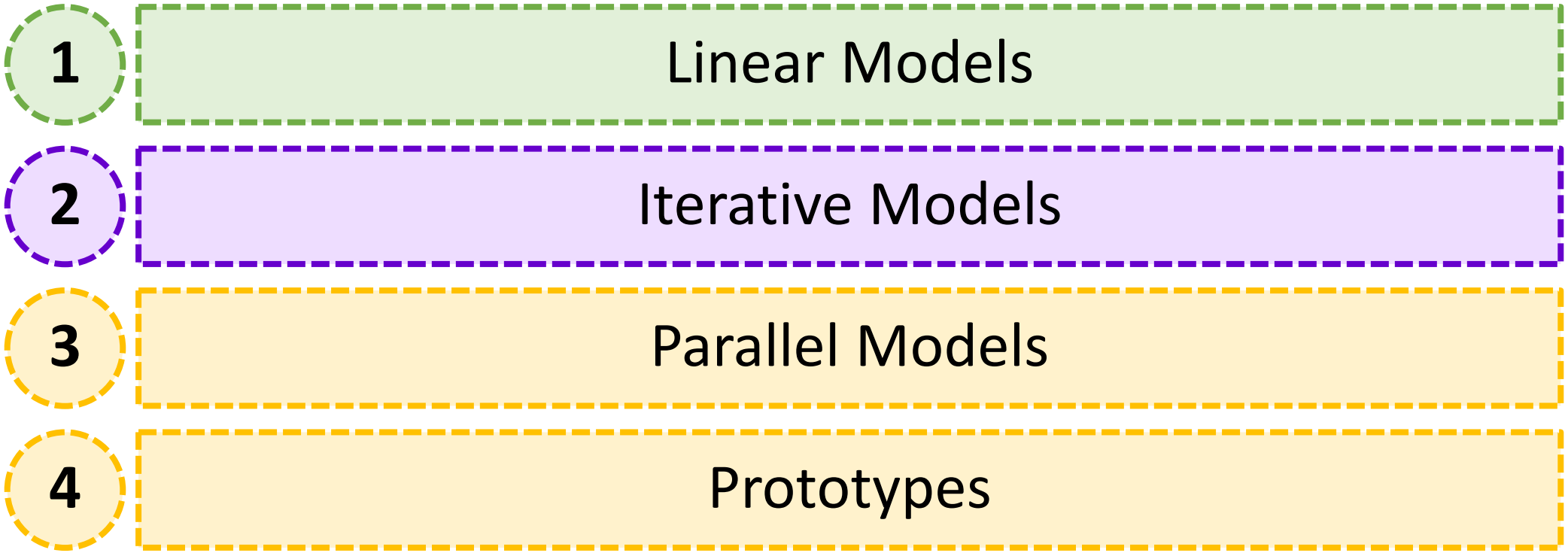
V-Model: Benefits and Drawbacks

The Sawtooth model is also a linear process model, but unlike the Waterfall and V models, the client is involved to review intermediate prototypes of the product during the process

Major flaw: Sawtooth model is still a linear process, so there is a limit to incorporating anything more than incremental changes.

Having the client involved earlier in the process increases the likelihood that the product will meet the client's needs

Session Roadmap



Iterative Models

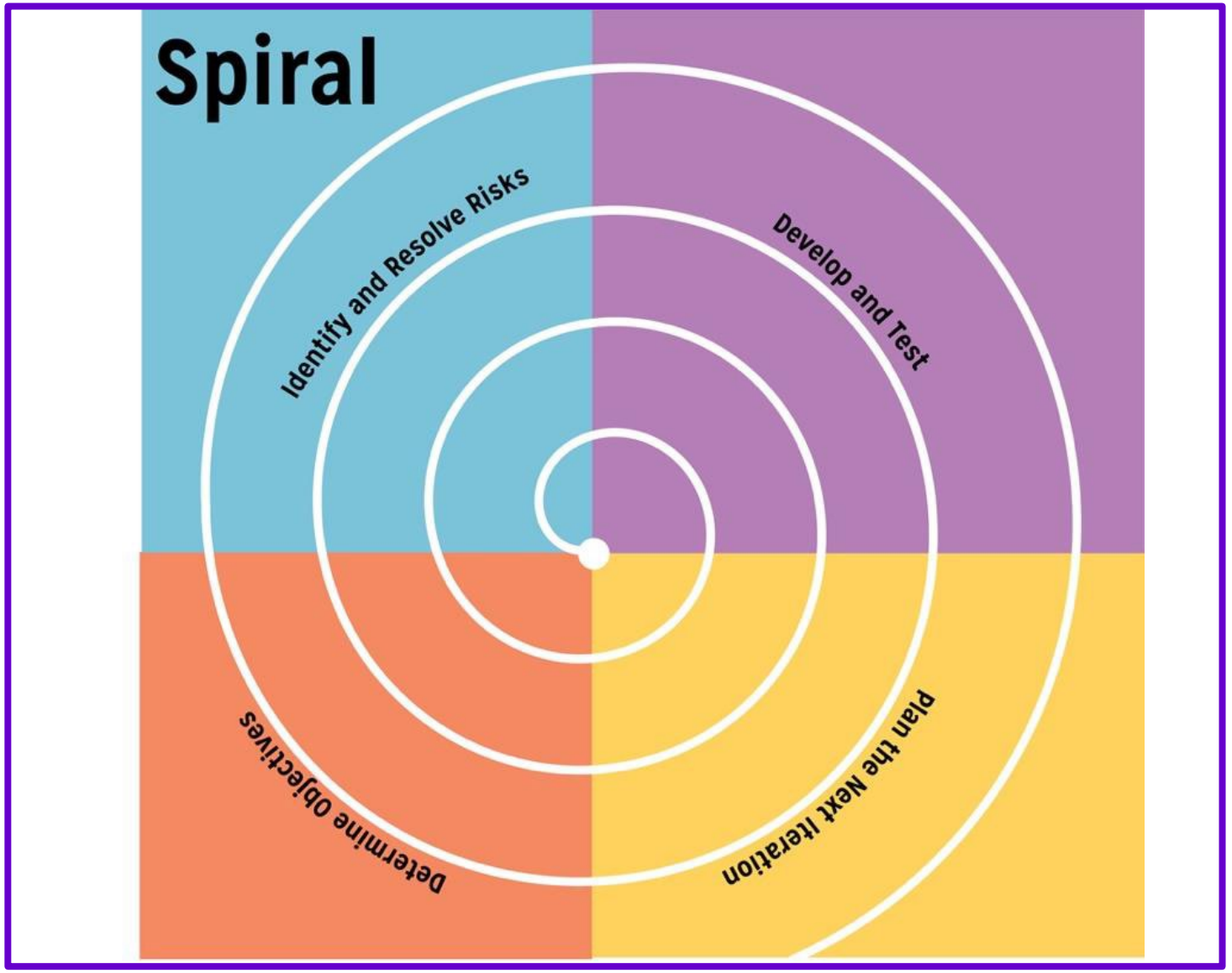
Iterative process models differ from linear process models in that they are designed for repeating stages of the process. That is, they are iterative or cyclical in structure

Iterations allow for client feedback to be incorporated within the process as being the norm, not the exception.

Iterative process models are readily amenable to Agile practices, yet they also embody sequential portions reminiscent of linear process models.

Iterative Models: Spiral Model

Each subsequent iteration has the sequence of the four phases revisited, and each iteration results in a product **prototype**.



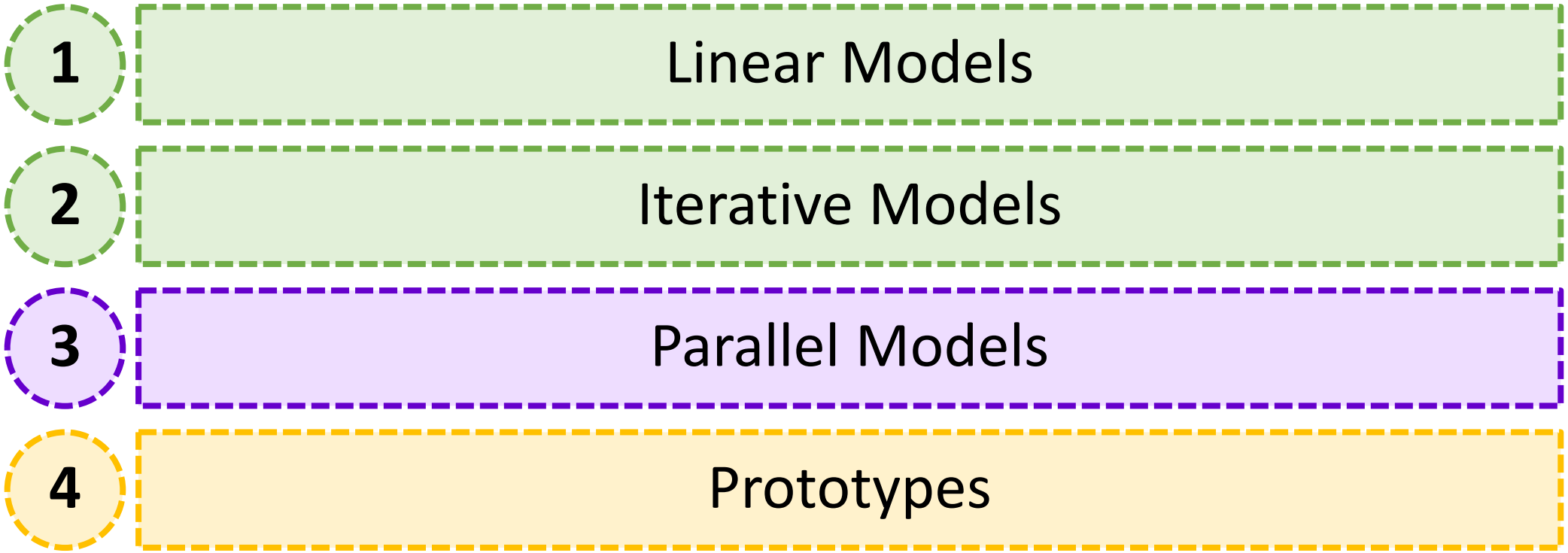
Spiral Model: Benefits and Drawbacks

Early iterations lead to product ideas and concepts, while later iterations lead to working software prototypes.

Estimating work can be more difficult, depending on the duration of the iteration cycle in the Spiral model. Not all organizations will have the years of experience, data, and technical expertise available for estimation and risk assessment

Allows the development team to review their product with the client to gather feedback and improve the product.

Session Roadmap



Parallel Models

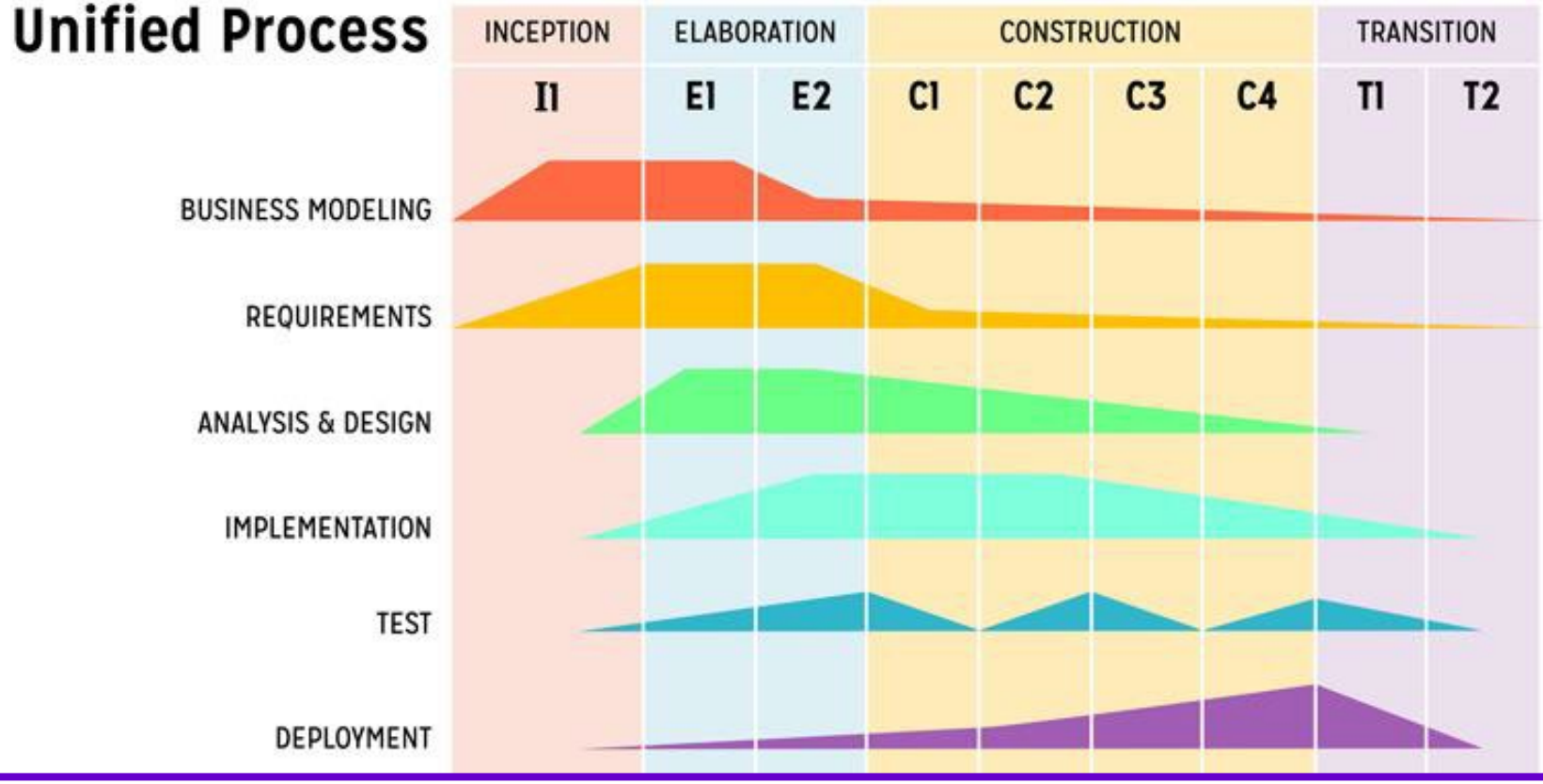
Parallel processes use a similar style of iteration—products are built in iterations—however, parallel processes allow for more concurrency of activities and tasks.

Iterations allow for client feedback to be incorporated within the process as being the norm, not the exception.

Iterative process models are readily amenable to Agile practices, yet they also embody sequential portions reminiscent of linear process models.

Parallel Models: Unified Process Model

The **Unified Process** basic structure has sequential phases within a repeatable cycle.



Phases within the Unified Process Model:
Inception phase

This first phase is meant to be short—enough time to establish a strong enough business case and financial reason to continue on to the next phases and make the product.

To do this, the *inception phase* typically calls for the creation of **basic use cases**, which outline the main user interactions with the product. Also defined are the project's **scope** and potential project **risks**.

Phases within the Unified Process Model: Elaboration phase

The goal of the elaboration phase is to create design models and prototypes of the product as well as to address risks

They also develop key requirements and architecture documentation, such as **use case diagrams**, and high-level **class diagrams.** At the end of the elaboration phase, **developers deliver a plan for development** in the next phase.

Phases within the Unified Process Model: Construction phase

The construction phase has iterations and focuses on building upon the previous work so far. This is where the software product begins to take shape.

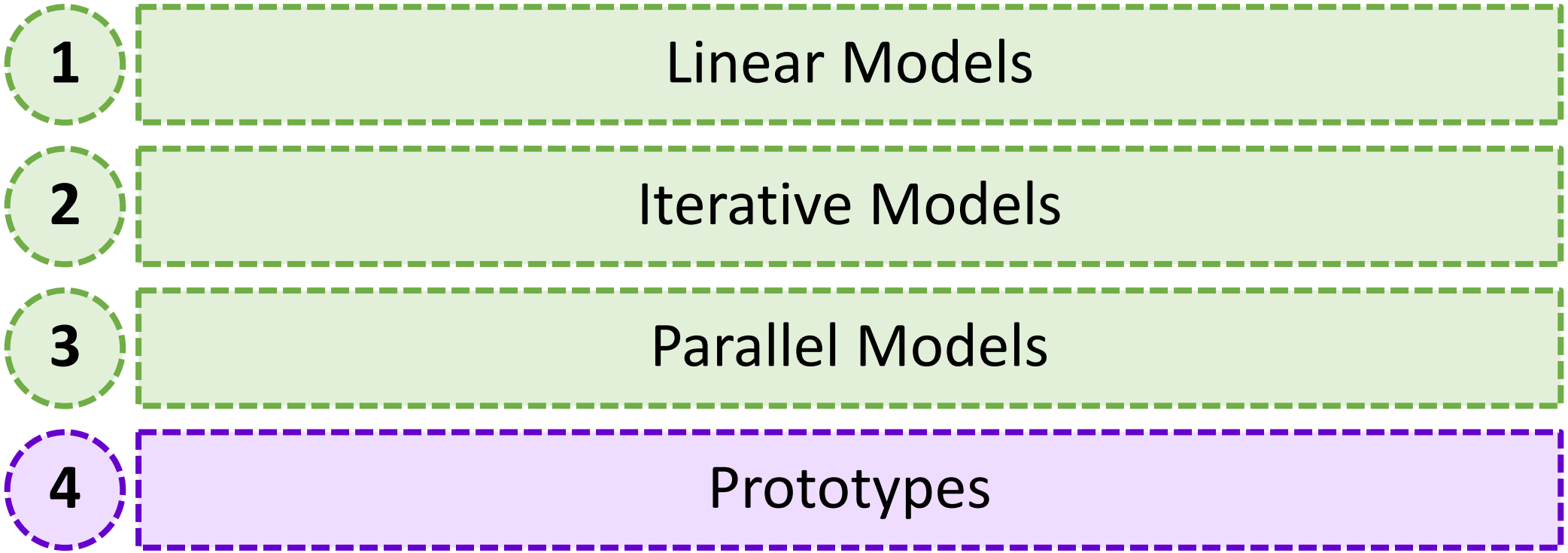
In the construction phase, **full use cases** are developed to drive product development. The product is built iteratively throughout the construction phase **until it is ready to be released**.

Phases within the Unified Process Model: **Transition phase**

The *transition phase* reveals how well the product design measures up against users' needs.

Upon completing the transition phase, it is possible to cycle back through the phases of the Unified Process again. For example, there may be a need to create further releases of the product or to incorporate user feedback as a means of influencing the plans for later development.

Session Roadmap



Prototypes: Illustrative prototype

This is the most basic of all prototypes. They could be drawings on a napkin, a set of slideshow slides, or even a couple index cards with components drawn out.

The essence of an *illustrative prototype* is to get a basic idea down.

Illustrative prototypes can give the development team a common idea off of which to base their work, or to help them get a system's "look and feel" right without investing much time or money into developing a product

Prototypes: Exploratory Prototype

By building working code, the development team can explore what is feasible—fully expecting to throw the work out after learning from the prototype.

Exploratory prototyping is expensive in terms of consuming resources, but it is used because it is better and less costly than finding out later in the process that a product solution just cannot work

The usual motivation behind exploratory prototyping is that the product developers want to study the feasibility of a product idea.

Prototypes: Throwaway Prototype

The first version of almost any product is bound to have various problems. So, why not just build a second, better, version from scratch and toss away the first?

A common feature of illustrated, exploratory, and throwaway prototypes is that none of the effort to build the prototypes will end up in the final version of the product.

This affords the chance to build the software product on a more robust second-generation architecture, rather than a first-generation system with patches and fixes.

Prototypes: Incremental Prototype

When a product is built and released in increments, it is an *incremental prototype*. Incremental prototyping works in stages, based on a **triage** system.

Priorities for a software product's features are based on three categories: "**must do**," "**should do**," and what "**could do**." Core features are assigned the highest priority—must do. All the non-critical supporting features are "should do" items. Remaining extraneous features are the lowest priority—could do.

Prototypes: Evolutionary Prototype

In evolutionary prototyping, the first-generation prototype has all the features of the product, even though some features may need to “evolve” or be refined.

A comparable first-generation incremental prototype has Both incremental and evolutionary prototyping are ways to make working software that can be shown at regular intervals to gain further feedback.

Course References