CCS|CSE|CIS1072- Introduction to Software Engineering

SOFTWARE PROCESS

Romenia Silva
romaniasilva@sjp.ac.lk

Software Process

A structured set of activities required to develop a software system.

Key Process Activities

01 **Specification** – defining what the system should do;

02 **Design and implementation** – defining the organization of the system and implementing the system;

03 **Validation** – checking that it does what the customer wants;

04 **Evolution** – changing the system in response to changing customer needs.

A **software process model** is an abstract representation of a process.

It presents a description of a process from some particular perspective.

Software Process Description

Process descriptions may also include:

- Products, which are the outcomes of a process activity;
- Roles, which reflect the responsibilities of the people involved in the process;
- Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

Plan-Driven and Agile Processes

Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.

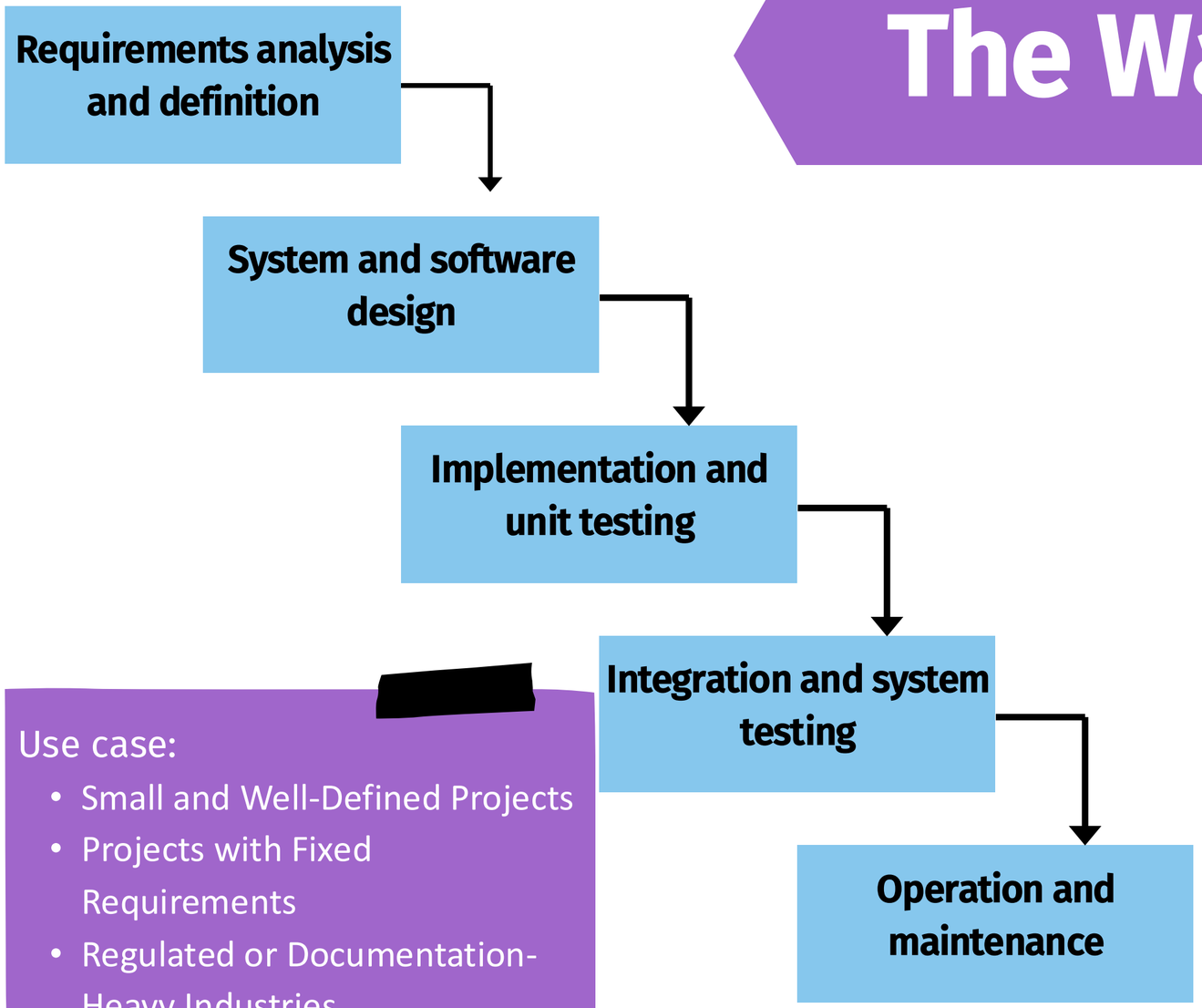
In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.

In practice, most practical processes include elements of both plan-driven and agile approaches.

There are no right or wrong software processes.

Software Process Models

The Waterfall Model



Use case:

- Small and Well-Defined Projects
- Projects with Fixed Requirements
- Regulated or Documentation-Heavy Industries

Plan-driven model. Separate and distinct phases of specification and development.

The **main drawback** of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase

The Waterfall model is suitable for large systems engineering projects where requirements are stable and extensive coordination and documentation are required. *Its plan-driven nature can help manage development across multiple sites.*

Problems of Waterfall Model

1. Real projects rarely follow the sequential work flow that the model proposes.
2. It is often difficult for the customer to state all requirements explicitly at the beginning of most projects.
3. The customer must have patience because a working version of the program(s) will not be available until late in the project time span.
4. Major blunders may not be detected until the working program is reviewed.

Waterfall Model Pros and Cons

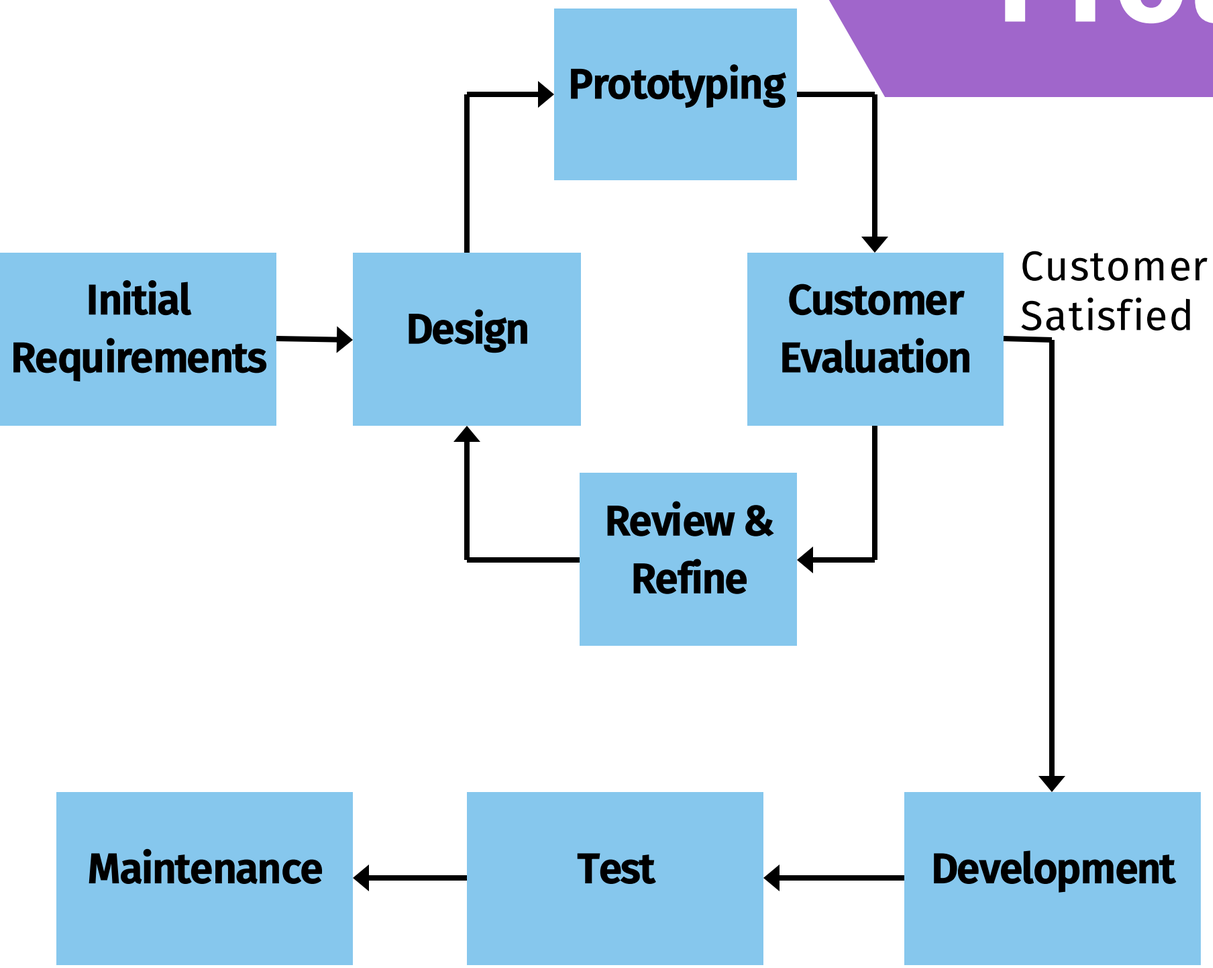
Pros

- It is easy to understand and plan.
- It works for well-understood small projects.
- Analysis and testing are straightforward.

Cons

- It does not accommodate change well.
- Testing occurs late in the process.
- Customer approval is at the end.

Prototyping Process Model



- Used when requirements are unclear or evolving.
- Helps both developers and stakeholders better understand system needs.
- Involves creating a quick design and working prototype to gather feedback.
- Iterative: prototype is refined through multiple cycles based on user input.
- Useful for Clarifying user interfaces, Exploring system behavior, Refining functional requirements.

Problems of Prototyping Model

1. **False sense of completeness:** Stakeholders may mistake the prototype for the final product, ignoring incomplete or unstable architecture.
2. **Poor quality structure:** The underlying design may lack scalability, maintainability, or robustness due to quick, iterative builds.
3. **Implementation shortcuts:** Developers might take temporary shortcuts just to make it work — these compromises can become permanent if not revisited.
4. **Risk of evolving a flawed system:** If early design flaws or poor decisions aren't corrected, they may persist and degrade the final product.

Prototype Model Pros and Cons

Pros

- There is a reduced impact of requirement changes.
- The customer is involved early and often.
- It works well for small projects.
- There is reduced likelihood of product rejection.

Cons

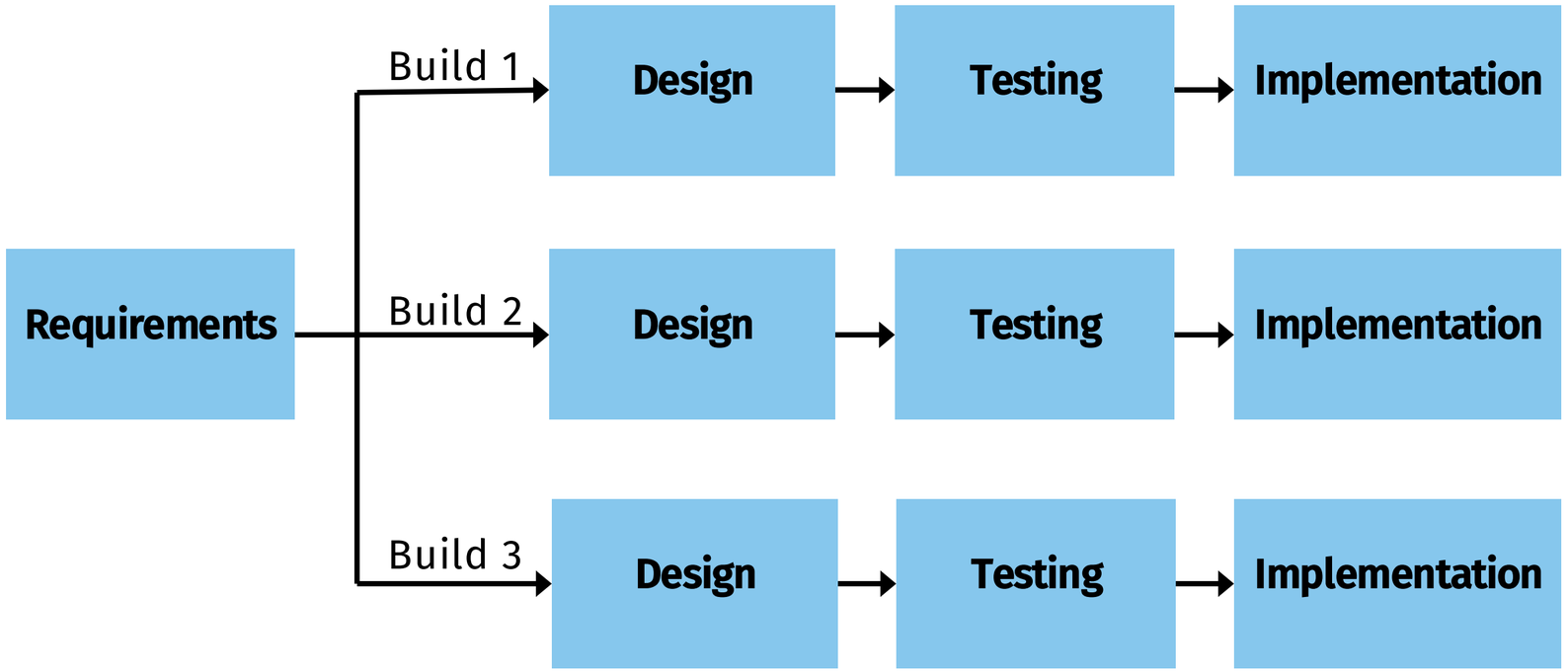
- Customer involvement may cause delays.
- There may be a temptation to “ship” a prototype.
- Work is lost in a throwaway prototype.
- It is hard to plan and manage.

Evolutionary Process Model

Why Evolutionary Models?

- Software requirements often change over time.
- Sometimes we need a quick version to meet business deadlines.
- Users give better feedback after seeing something real.
- Software is not built in one go – it evolves.

Incremental Model



Pros: Early delivery, manageable chunks, easier testing

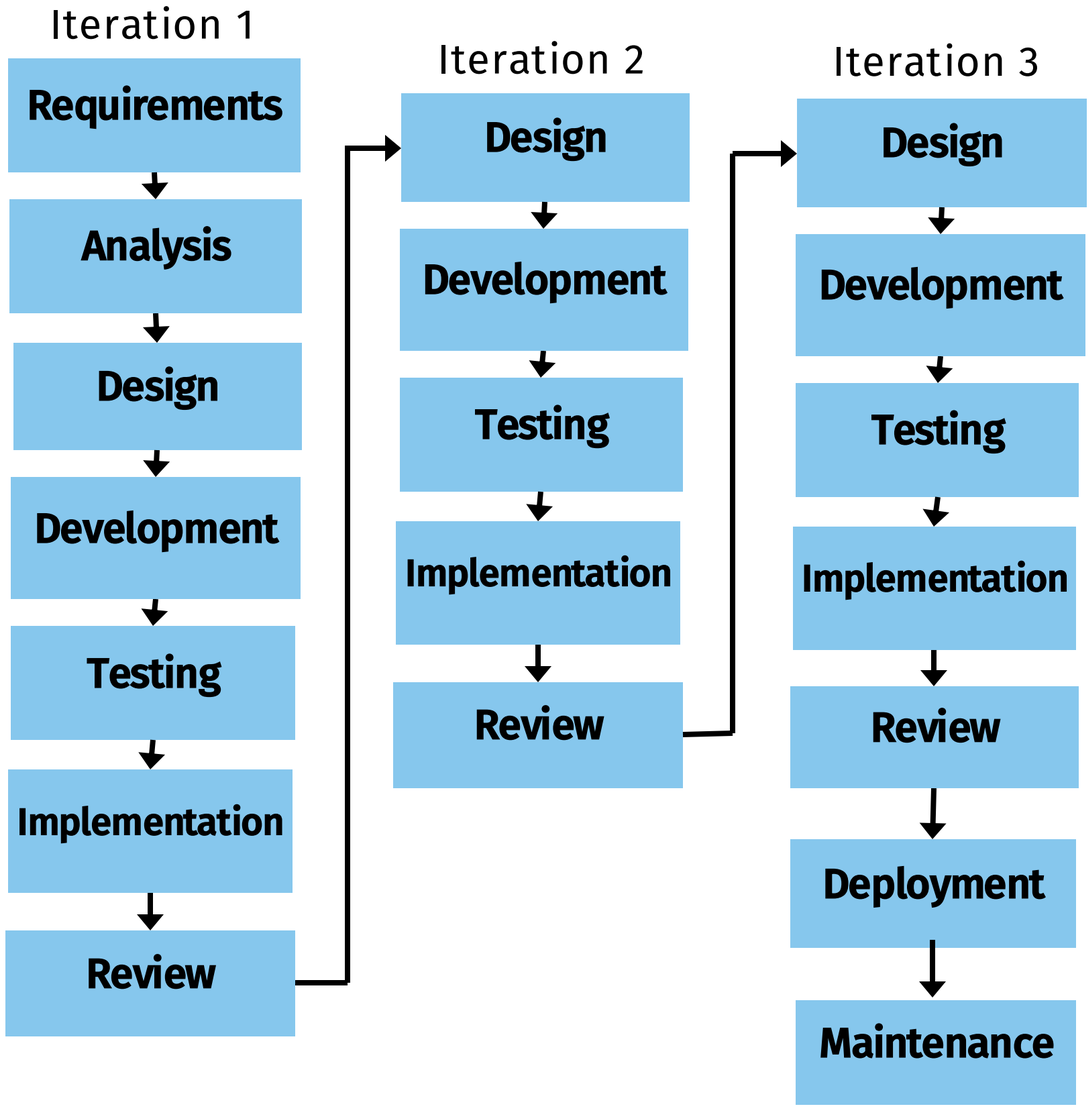
Cons: Requires clear planning of increments

Example:
1st release – Login & View
2nd release – Search & Cart
3rd release – Payment

Build software in parts (increments)

- Each increment adds new functionality.
- Earlier parts can be used while later parts are being developed.

Iterative Model



Build a simple version first, then improve it in cycles

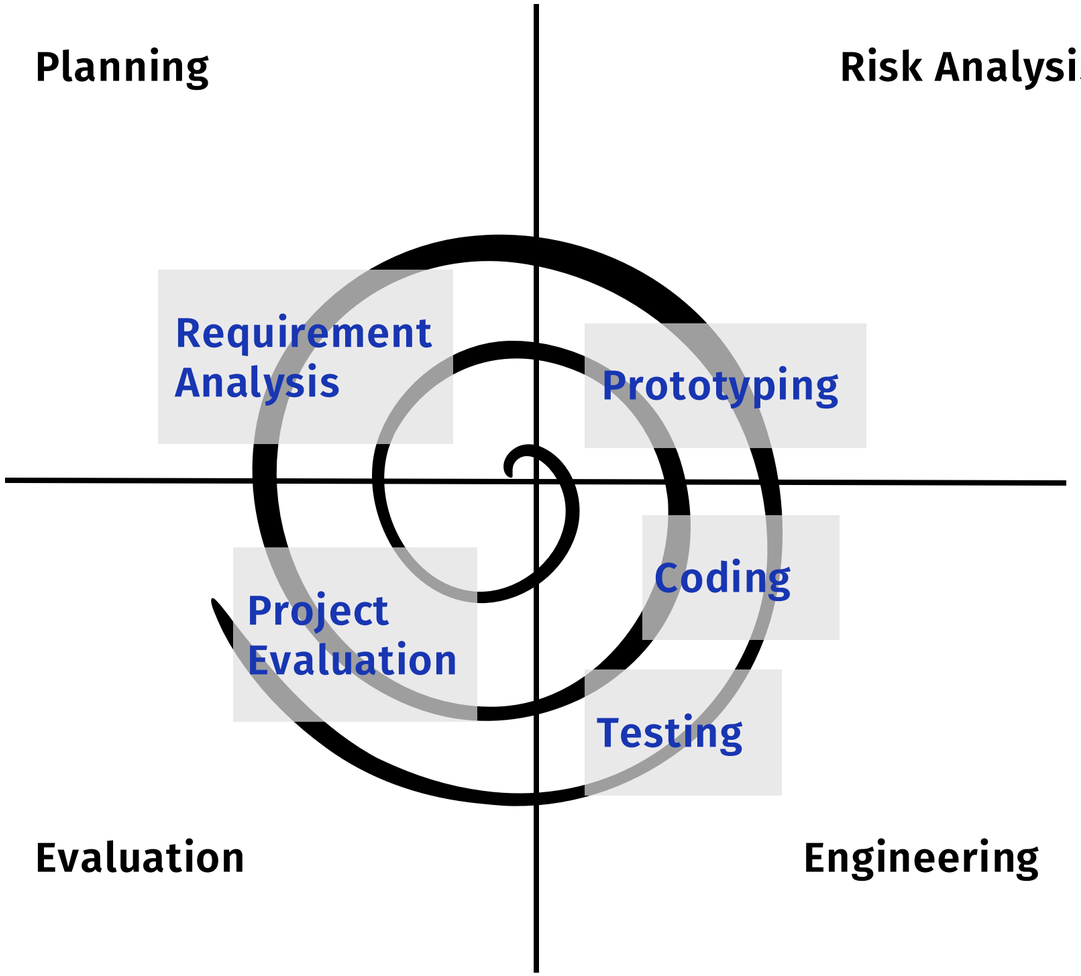
- Each iteration refines existing components.
- Good when requirements are unclear or evolving.

Pros: Continuous improvement, user feedback

Cons: May need frequent rework

Example:
1st version – Basic editor
2nd version – Add undo/redo
3rd version – Add formatting

Spiral Model



Combine iteration with risk management

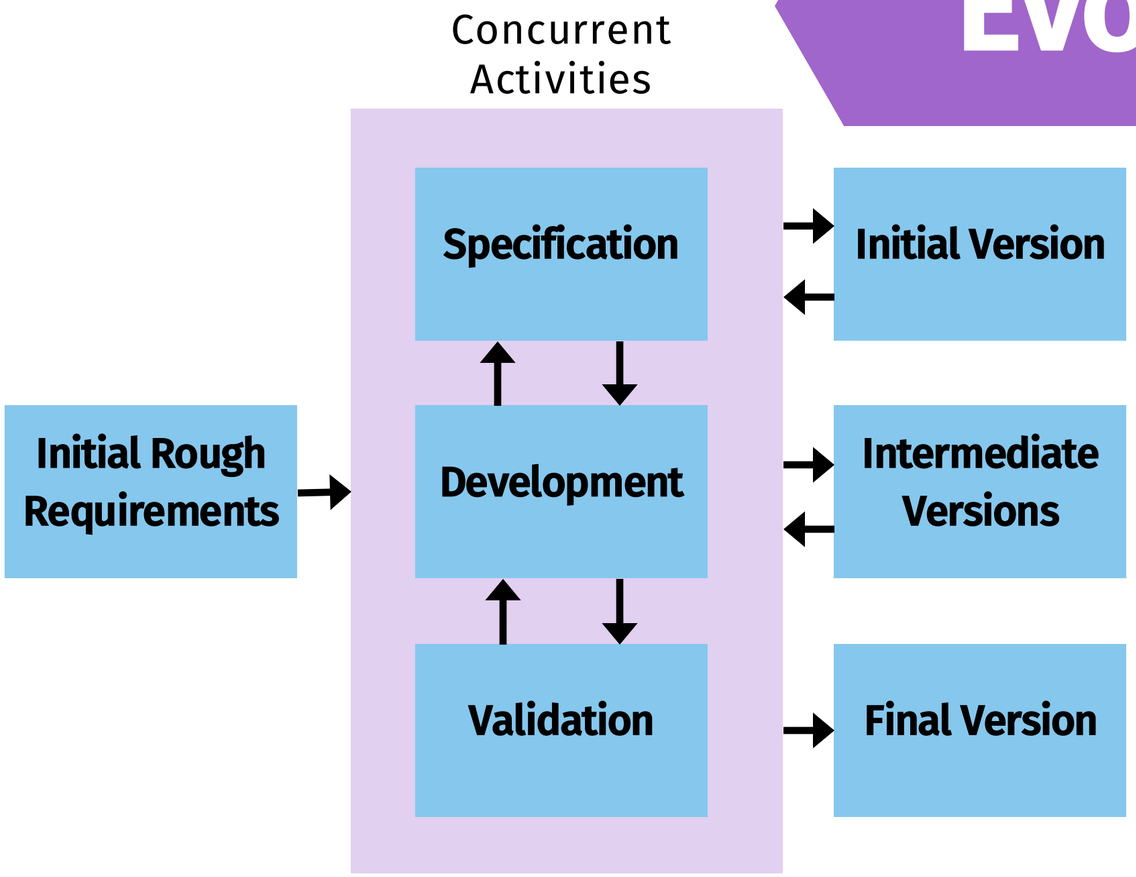
- Each loop = one phase (Planning → Risk → Development → Evaluation)
- Refines system with each spiral.

Pros: Good for large/risky projects, handles changing requirements

Cons: Complex, needs skilled risk assessment

Use case:
Government systems
Space software
Critical systems

Evolutionary Process Model



- A software development approach where the system is built through repeated refinements and incremental improvements.
- Combines principles of Incremental + Iterative + Spiral models
- System evolves with each cycle
- Delivers early versions, gathers feedback, improves further

When to Use Evolutionary Models

- Requirements are not fully known
- There's time pressure to release early
- High risk or complex systems
- Need for user feedback and adaptability

Evolutionary Model Pros and Cons

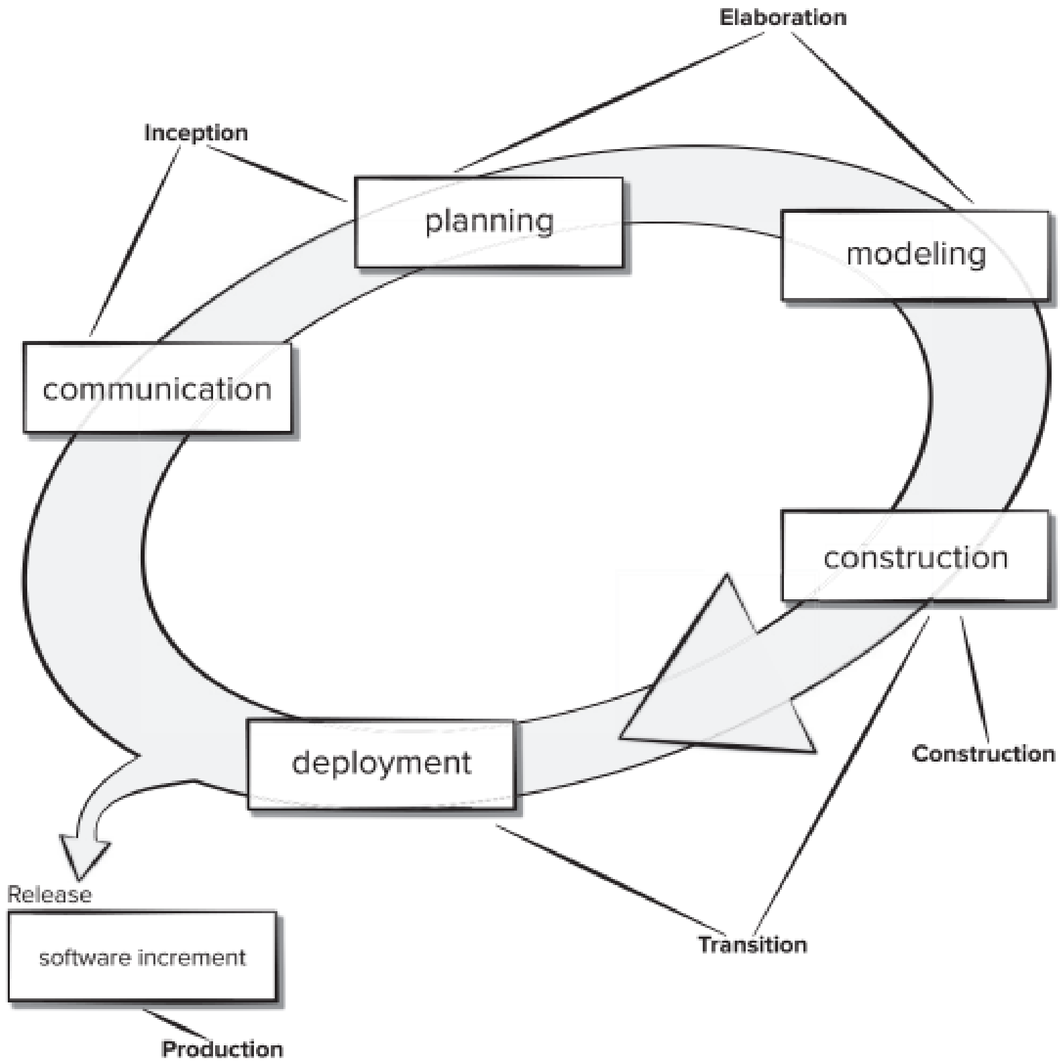
Pros

- Flexible to changing requirements
- Delivers usable software early
- Encourages customer feedback
- Reduces project risk
- Improves understanding of requirements

Cons

- Requires careful planning and management
- High customer involvement needed
- May increase overall cost
- Risk of poor architecture
- Time-consuming if not controlled

Unified Process Model



- **The Unified Process (UP) blends strengths of traditional and agile models for modern software development.**
- **Emphasizes:**
  - Customer communication using use cases
  - Strong software architecture (clarity, flexibility, reuse)
  - Iterative and incremental development (evolutionary approach)
- **OOP-based**: Built around object-oriented principles to structure software as modular, reusable components.
- **Supported by UML (Unified Modeling Language):**
- **Industry-standard for modeling object-oriented systems**
- **Used for requirements, design, and implementation modeling**.

UP Phases and Activities

Phase	Main Focus
Inception	- Understand the scope of the system - Identify key use cases and risks
Elaboration	- Refine use cases - Define architecture (5 views: use case, analysis, design, implementation, deployment)
Construction	- Build core components - Implement and test features iteratively
Transition	- Beta testing with users - Fix defects and finalize release
Production	- Deploy the system - Provide support and collect feedback for improvements

Unified Process Model Pros and Cons

Pros

- Structured yet flexible.
- Encourages early risk reduction and feedback.
- Promotes reusability and scalability.
- Good alignment with object-oriented design and UML.

Cons

- Can be complex for small or simple projects.
- Requires skilled team members.
- Needs proper management to avoid overlapping confusion.

Activity: Match the Model

Column A – Process Models

1.Waterfall Model
2.Prototyping Model
3.Incremental Model
4.Iterative Model
5.Spiral Model
6.Evolutionary Model
7.Unified Process Model

Column B – Characteristics

A. Develop the system through repeated cycles of improvement.
B. Combines waterfall structure with risk assessment and prototyping.
C. Software is developed and delivered in small, complete parts.
D. Structured, sequential development process; changes are difficult.
E. Combines the best of traditional models and agile principles; use case driven.
F. Develops a quick working system to gather feedback before building the final one.
G. A broad approach that allows the system to evolve over time with multiple releases.

That's a wrap!

Thank you!

May 18, 2026