KAIST

Spring 2025

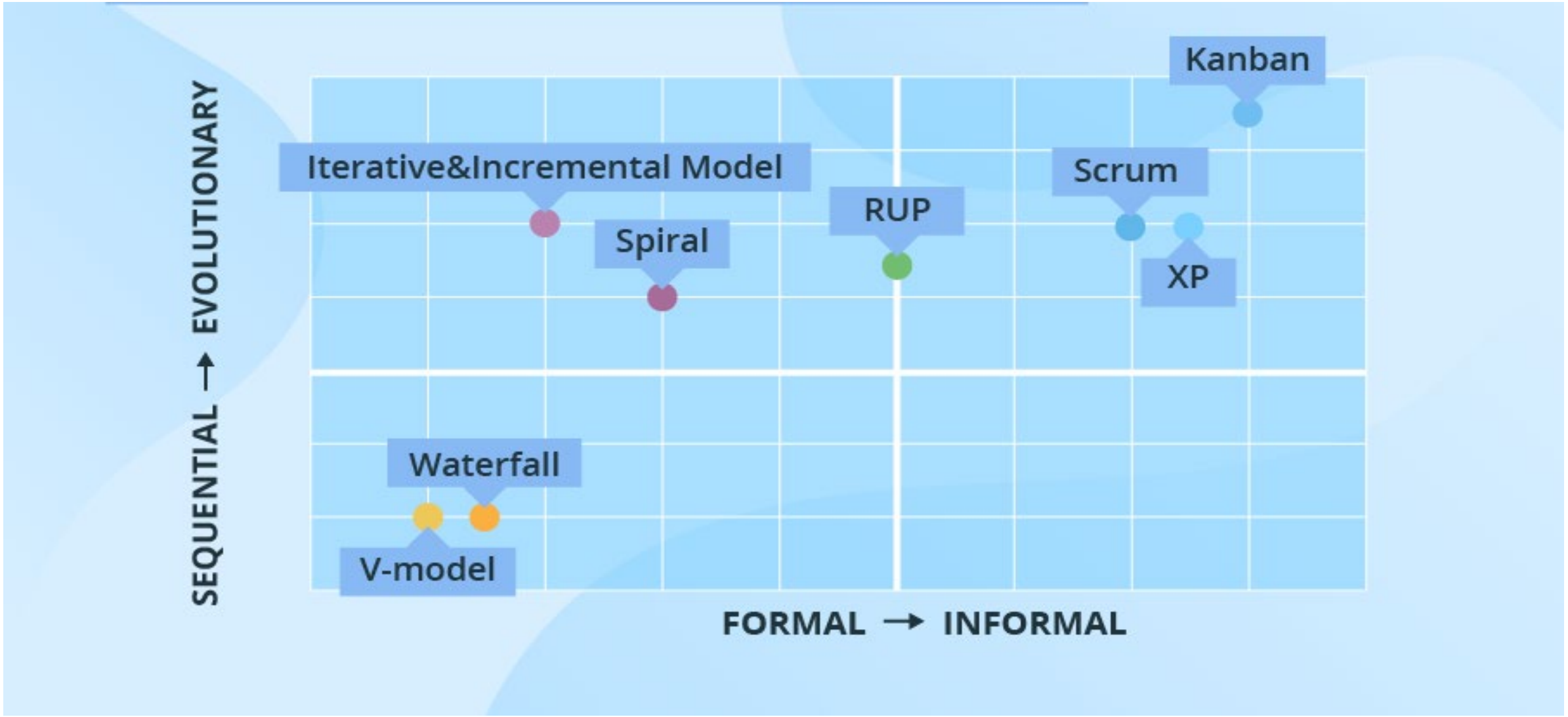
CS30500:
Introduction to Software Engineering

Prof. In-Young Ko

School of Computing

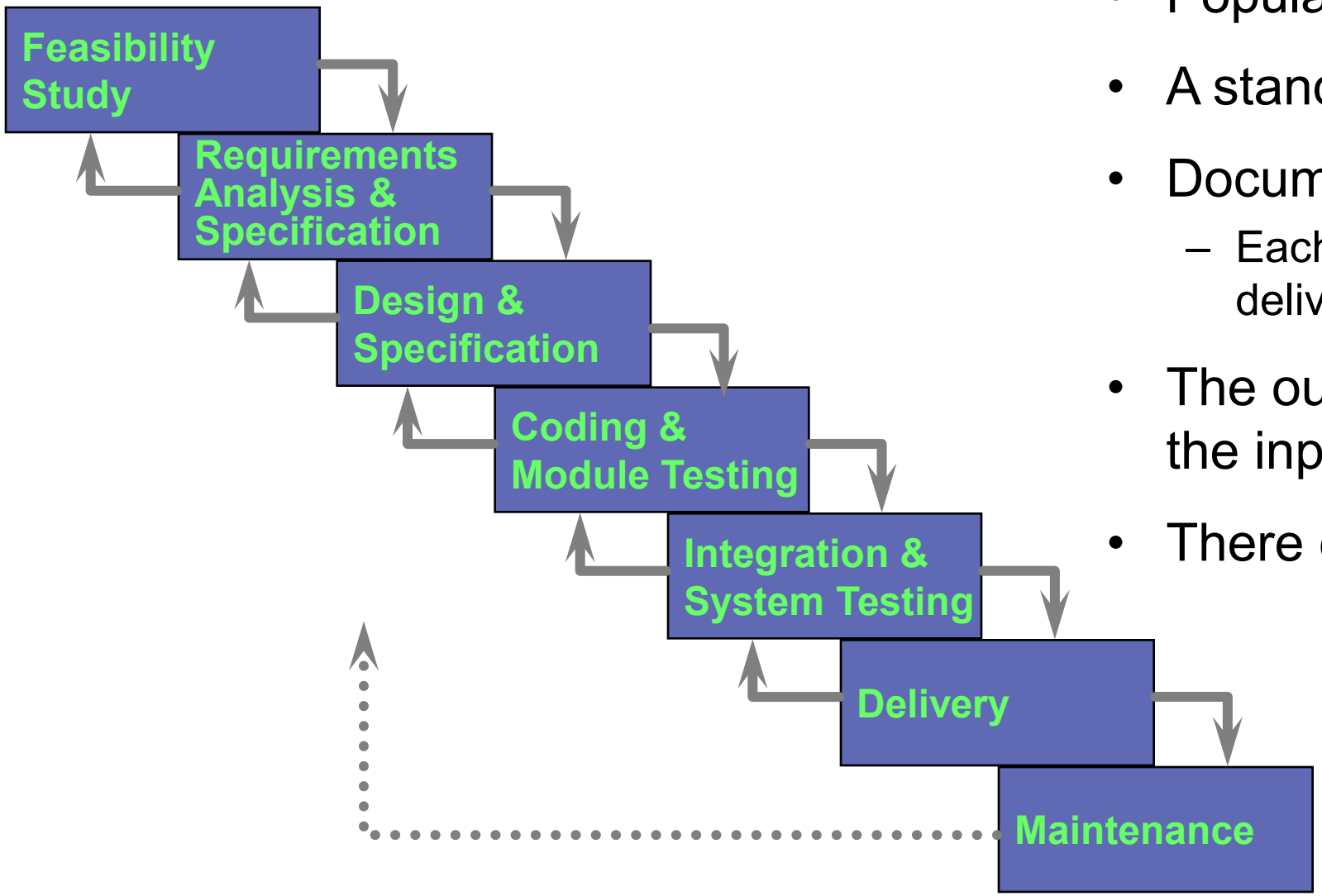
TRADITIONAL SOFTWARE PROCESS MODELS

Software Development Lifecycle Models



[Shi19] Boris Shiklo, 8 Software Development Models: Sliced, Diced and Organized in Charts (https://www.scnsoft.com/blog/software-development-models)

Waterfall Model



- Popular in 1970's
- A standard industrial practice
- Document-oriented
  - Each stage produces concrete deliverables (documents)
- The output of one phase constitutes the input to the next
- There exist many variants

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Waterfall Model – Use Cases

- Simple small and mid-size projects with clearly defined and unchanging requirements
- Projects with the need for stricter control, predictable budget and timeline
- Projects that must adhere to multiple rules and regulations
- Projects where well-known technology and tools are used

[Shi19]

Waterfall Model – Feasibility Study

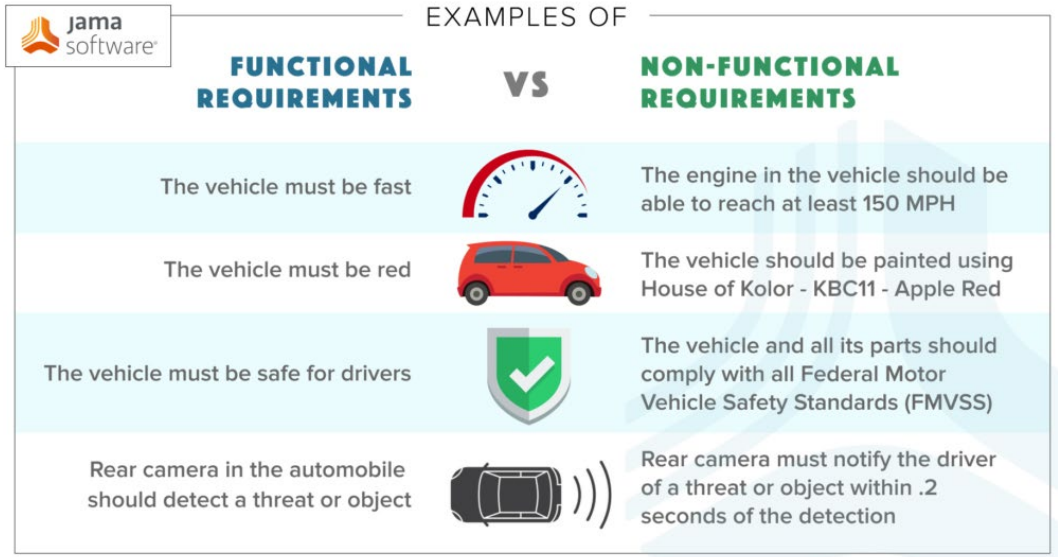
- Evaluates the costs and benefits of the proposed solution
- Analyzes the problem, at least at the global level
- Simulates the future development process
- Documents produced:
  - A definition of the problem
  - Alternative solutions and their expected benefits
  - Required resources, costs, and delivery dates in each

https://medium.com/@sadiamujtaba18/essentials-of-cost-benefit-analysis-in-business-4d3dc16083d1

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Waterfall Model – Requirements Analysis and Specification

- Identify the qualities required for the application
  - Functional and nonfunctional
- Must state what to do, not how to do
- Used by both customers and designers
- Separate functional requirements into three views:
  - Model of data
  - Model of function
  - Model of controls

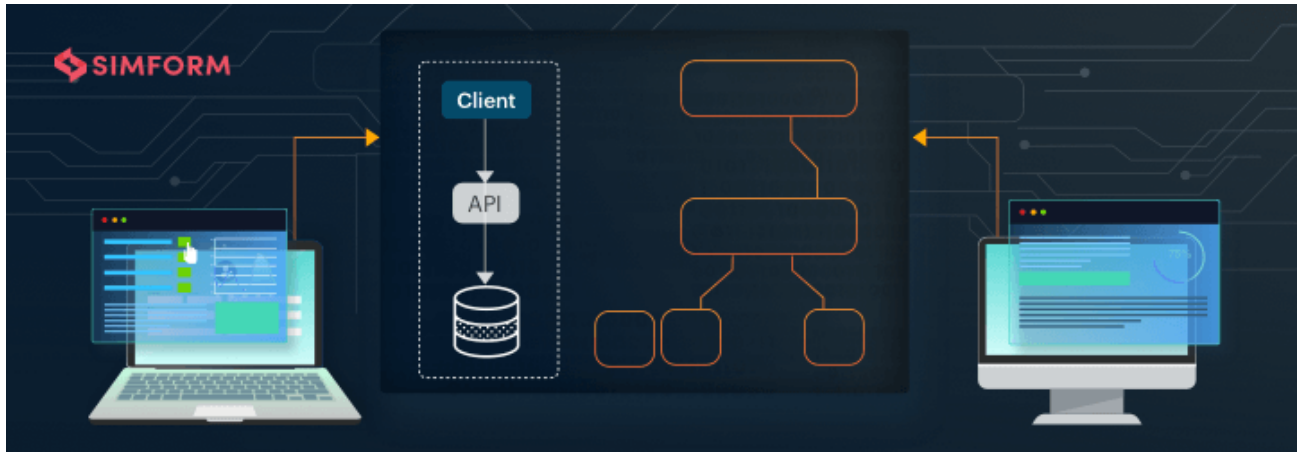


https://www.jamasoftware.com/requirements-management-guide/writing-requirements/functional-vs-non-functional-requirements

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Waterfall Model – Design and Specification

- Propose a solution to the problem
- Decompose the system into modules
  - Specify the relationships among modules
  - Design each module
- Can be divided into
  - High-level (preliminary, architectural)
  - Low-level (detailed)
  - User interface

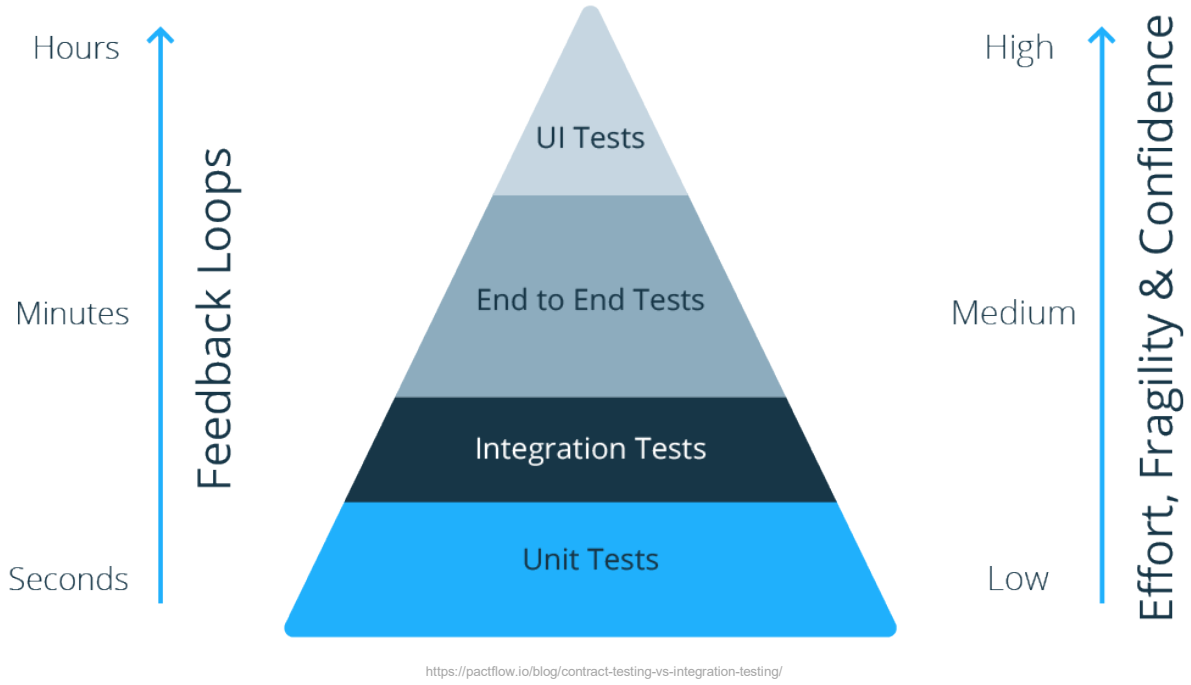


https://www.simform.com/blog/software-architecture-patterns/

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Waterfall Model – Coding and Testing

- Ideally transform of design into code
- Testing
  - Unit (module) testing
  - Integration testing
  - System testing
  - Acceptance testing
  - Etc….



https://pactflow.io/blog/contract-testing-vs-integration-testing/

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Waterfall Model – Delivery and Maintenance

- Types of maintenance
  - Corrective: 20%
  - Adaptive: 20%
  - Perfective: 60%
- Requirements analysis is a serious source of problems
- Many errors are not removed until after the system is delivered
- It is difficult to incorporate changes in the product

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Waterfall Model – Other Activities

- Documentation
- Verification
  - Monitor the quality of the application
  - Perform at the end of each phase
  - Methods
    - Review
    - Walk-through
    - Inspection
- Management
  - Tailoring the process
  - Defining policies
  - Dealing all resources affecting the process

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Characteristics of the Waterfall Model

- Linear
  - Development may proceed linearly from analysis to coding
- Rigid
  - The results of each phase are frozen before proceeding to the next
- Monolithic
  - All planning is oriented toward a single delivery date

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

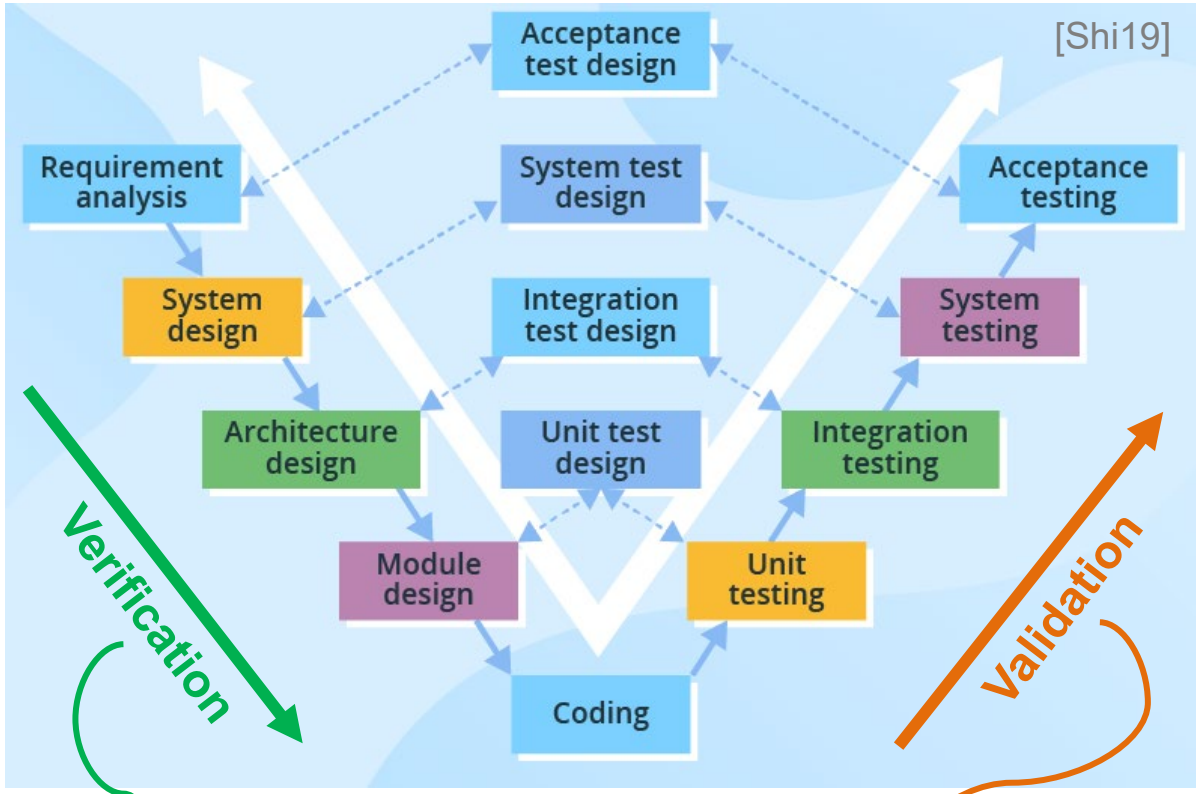
Pros and Cons of the Waterfall Model

- Pros
  - Enforced disciplined, planned and manageable approach
  - Product implementation is done after the objectives of doing so are well understood
- Cons
  - Difficult to estimate resources accurately
  - Verification of requirements specification by customer is not effective
  - Stakeholders often do not know the exact requirements
  - Does not stress the need for anticipating changes
  - Leads to a somewhat bureaucratic style of work

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

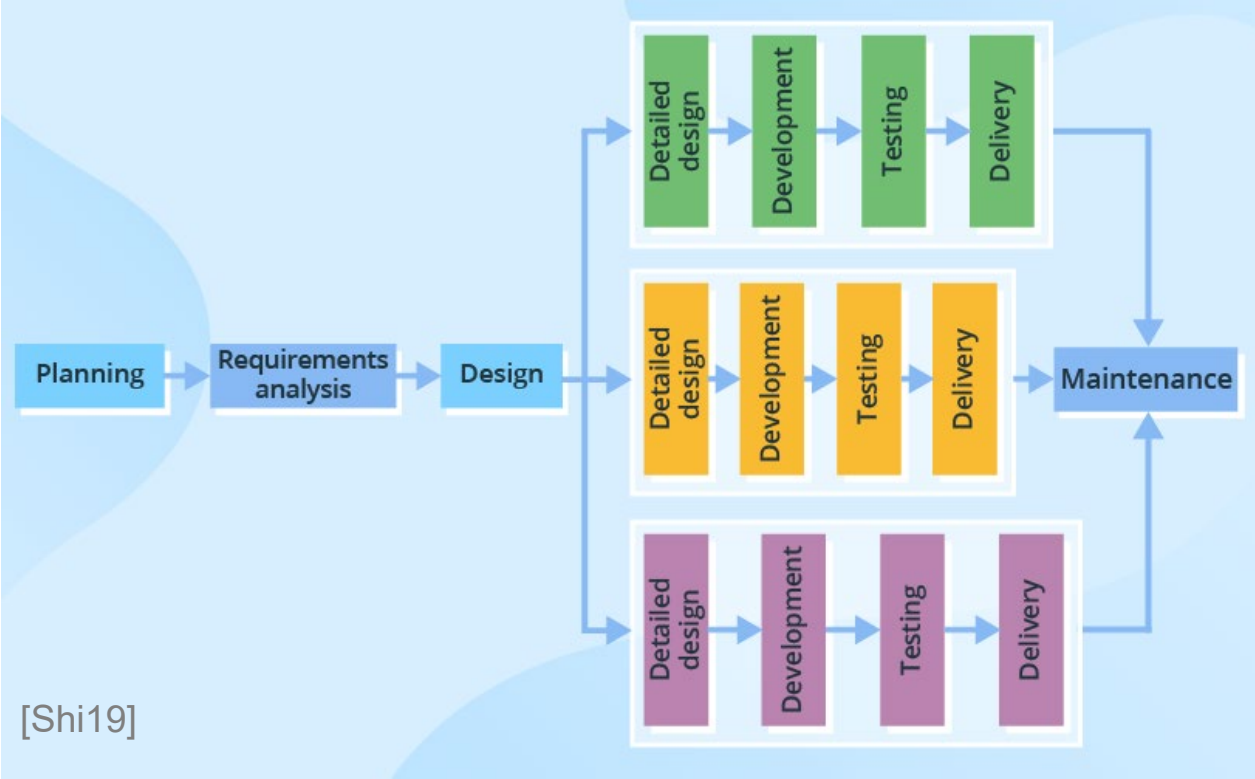
V-Model

- Linear model – each stage has a corresponding testing activity
- Exceptional quality control, expensive and time-consuming model
- Changes during development are expensive and difficult to implement
- Use cases:
  - Projects where failures and downtimes are unacceptable – e.g., Medical, aviation, and automobile software



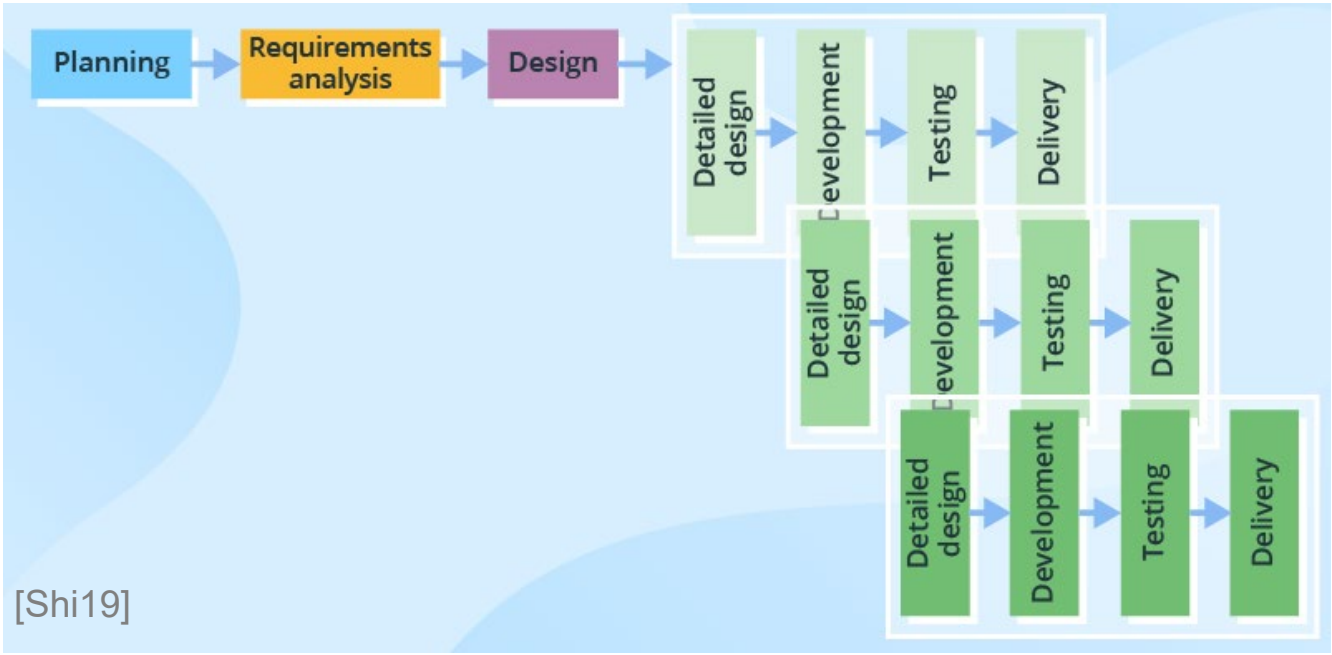
Incremental

- Split into several iterations
- New <u>software modules</u> are added in each iteration with no or little change in earlier added modules
  - Sequential and Parallel



Iterative

- Software changes on each iteration, evolves, and grows
- No need for a full specification from the project's start and small changes to requirements are possible in the course of development process
- Entails some customer involvement
- Use cases for Incremental and Iterative process
  - Large, mission-critical enterprise applications with loosely coupled parts

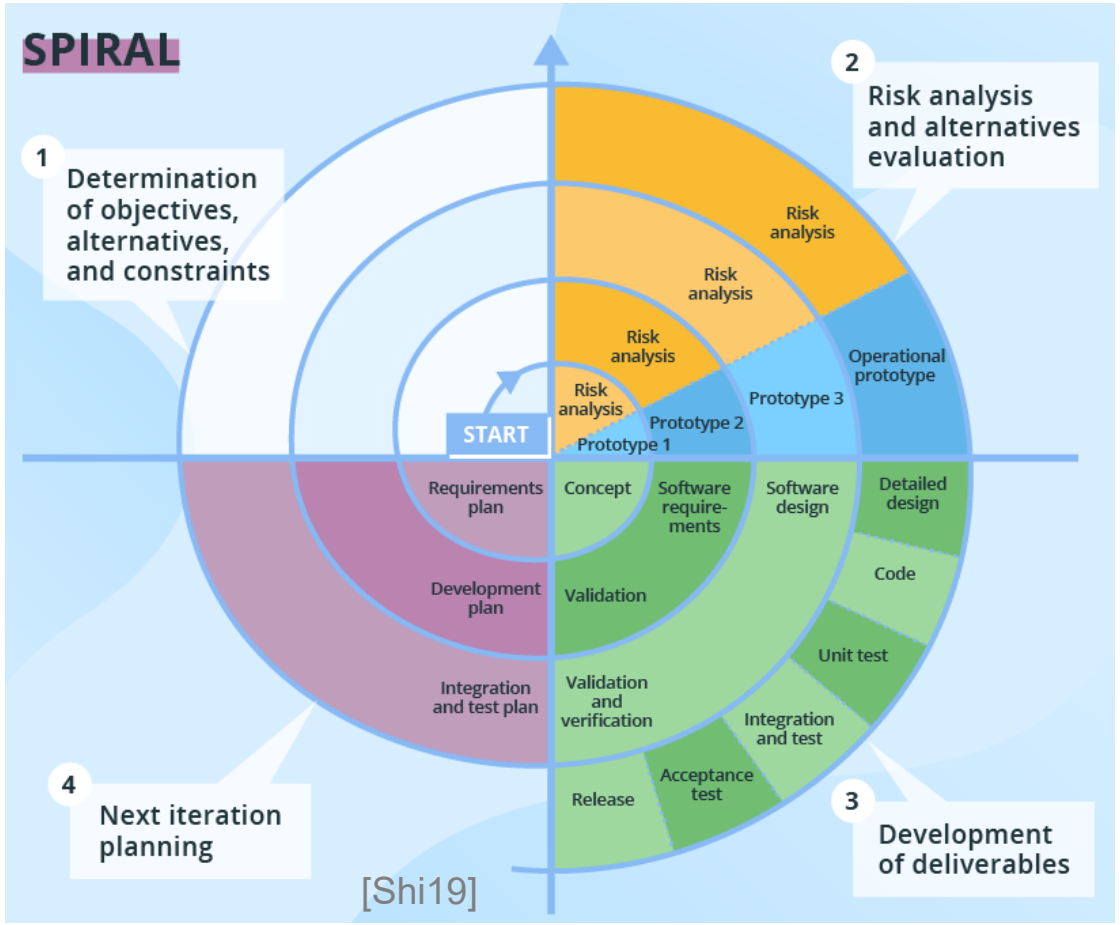


[Shi19]

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

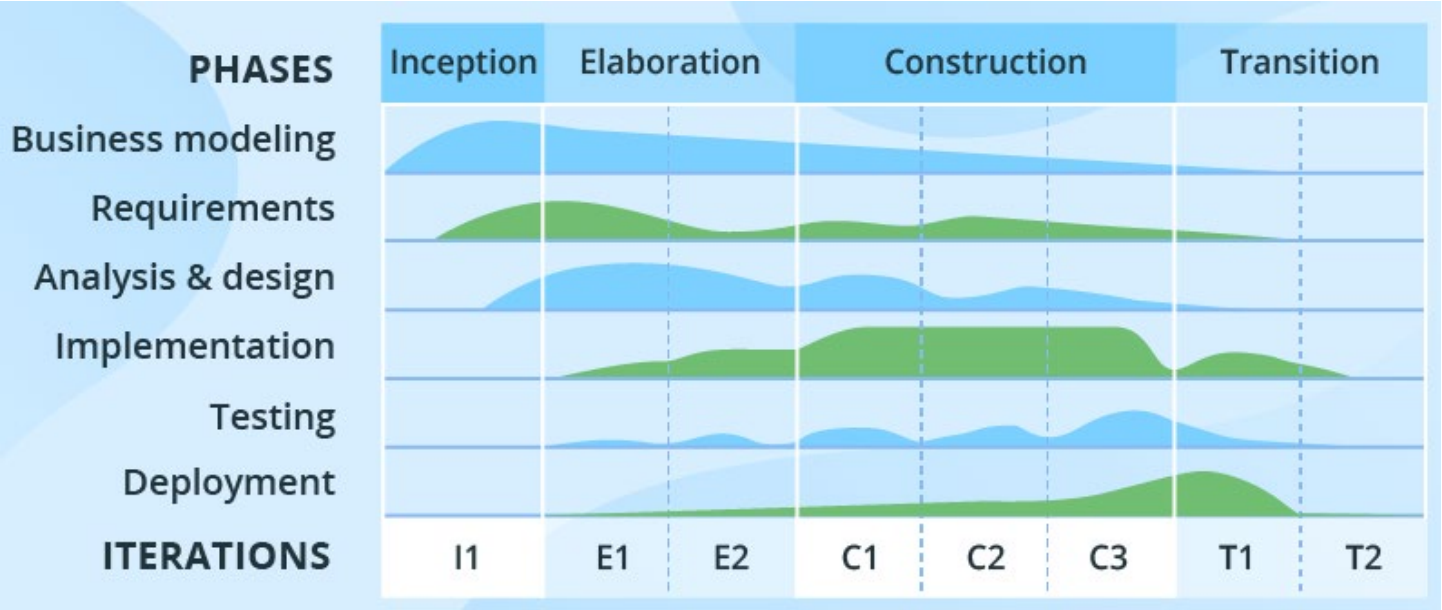
Spiral Model

- Developed by Prof. Barry Boehm (1986)
- Focus on through risk analysis
- A typical iteration lasts about 6 months
- Intensive customer involvement
- Use cases:
  - Projects with unclear business needs or too ambiguous/innovative requirements
  - Large and complicated
  - R&D or introduction of new service/product



Prof. Boehm's talk - https://youtu.be/DXgL62w11Gk (13:52-17:15)

Rational Unified Process (RUP)

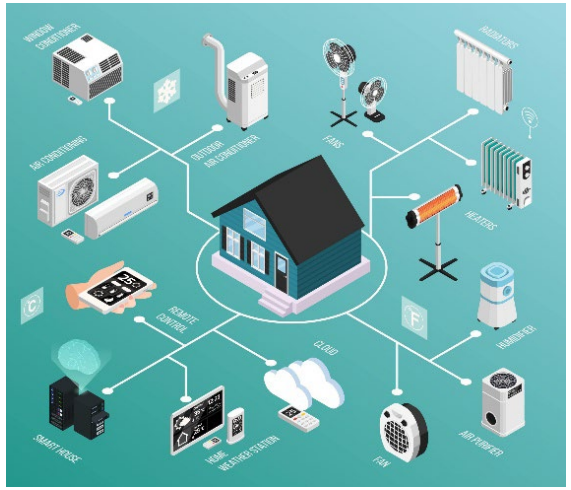


[Shi19]

- Combination of linear and iterative processes
- Consists of 4 phases: inception, elaboration, construction and transition
- All basic activities (requirements, design, etc.) of the development process are done in parallel with different intensity
- Use cases
  - Large and high-risk projects, especially use-case based development

Adopted from Prof. Doo-Hwan Bae's CS350 lecture material

Quick Overview of SafeHome



- The SafeHome company has developed an innovative HW box that implements wireless Internet (802.11n) connectivity in a very small form factor (the size of a matchbook).
- The idea is to use this technology to develop and market a comprehensive home automation product line.
  - This would provide
    - security functions
    - control over telephone answering machines
    - lights
    - heating
    - air conditioning
    - home entertainment devices.

- The first generation of the system will only focus on home security since that is a market the public readily understands.

Selecting a Process Model, Part 1(pg 28)

- **The scene:**
  - Meeting room for the software engineering group at CPI Corporation, a (fictional) company that makes consumer products for home and commercial use.
- **The players:**
  - Lee Warren, engineering manager;
  - Doug Miller, software engineering manager;
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member.
- **The conversation:**
- **Lee:** So let's recapitulate. I've spent some time discussing the *SafeHome* product line as we see it at the moment. No doubt, we've got a lot of work to do to simply define the thing, but I'd like you guys to begin thinking about how you're going to approach the software part of this project.

- **Doug:** Seems like we've been pretty disorganized in our approach to software in the past.
- **Ed:** I don't know, Doug. We always got product out the door.

Doug: True, but not without a lot of grief, and this project looks like it's bigger and more complex than anything we've done in the past.

- **Jamie:** Doesn't look that hard, but I agree ... our ad hoc approach to past projects won't work here, particularly if we have a very tight timeline.
- **Doug (smiling):** I want to be a bit more professional in our approach. I went to a short course last week and learned a lot about software engineering ... good stuff. We need a process here.

KAIST

- **Jamie (with a frown):** My job is to build computer programs, not push paper around
- **Doug:** Give it a chance before you go negative on me. Here's what I mean. [Doug proceeds to describe the process framework described in Chapter 2 and the prescriptive process models presented to this point.
- **Doug:** So anyway, it seems to me that a linear model is not for us ... assumes we have all requirements up front and knowing this place, that's not likely.
- **Vinod:** Yeah, and that RAD model sounds way too IT- oriented ... probably good for building an inventory control system or something, but it's just not right for *SafeHome*
- **Doug:** I agree.
- **Ed:** That prototyping approach seems OK. A lot like what we do here anyway.
- **Vinod:** That's a problem. I'm worried that it doesn't provide us with enough structure.
- **Doug:** Not to worry. We've got plenty of other options, and I want you guys to pick what's best for the team and best for the project.

Selecting a Process Model, Part 2 (pg31)

- The players:
  - Lee Warren: engineering manager
  - Doug Miller: SE manager
  - Ed and Vinod: members of the SE team

The conversation: (Doug describes evolutionary process options)

- Ed: Now I see something I like. An incremental approach makes sense and I really like the flow of that spiral model thing. That's keeping it real.
- Vinod: I agree. We deliver an increment, learn from customer feedback, replan, and then deliver another increment. It also fits into the nature of the product. We can have something on the market fast and then add functionality with each version, er, increment.
- Lee: Wait a minute, did you say that we regenerate the plan with each tour around the spiral, Doug? That's not so great, we need one plan, one schedule, and we've got to stick to it.
- Doug: That's old school thinking, Lee. Like Ed said, we've got to keep it real. I submit that it's better to tweak the plan as we learn more and as changes are requested. It's way more realistic. What's the point of a plan if it doesn't reflect reality?
- Lee (frowning): I suppose so, but senior management's not going to like this… they want a fixed plan.
- Doug (smiling): Then, you 'll have to reeducate them, buddy