Software Development Lifecycles

CSE 403 Software Engineering

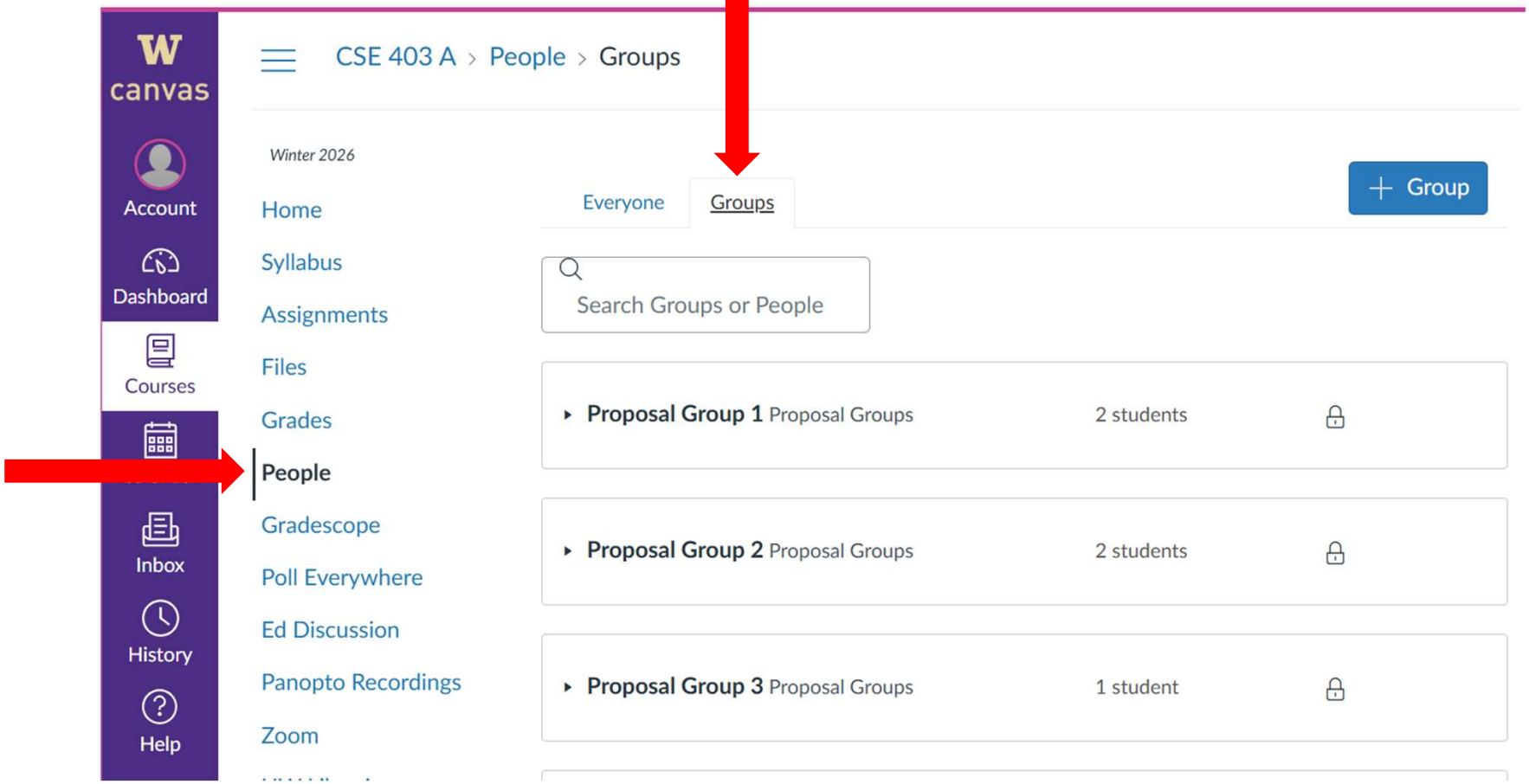
Winter 2026

Today's Outline

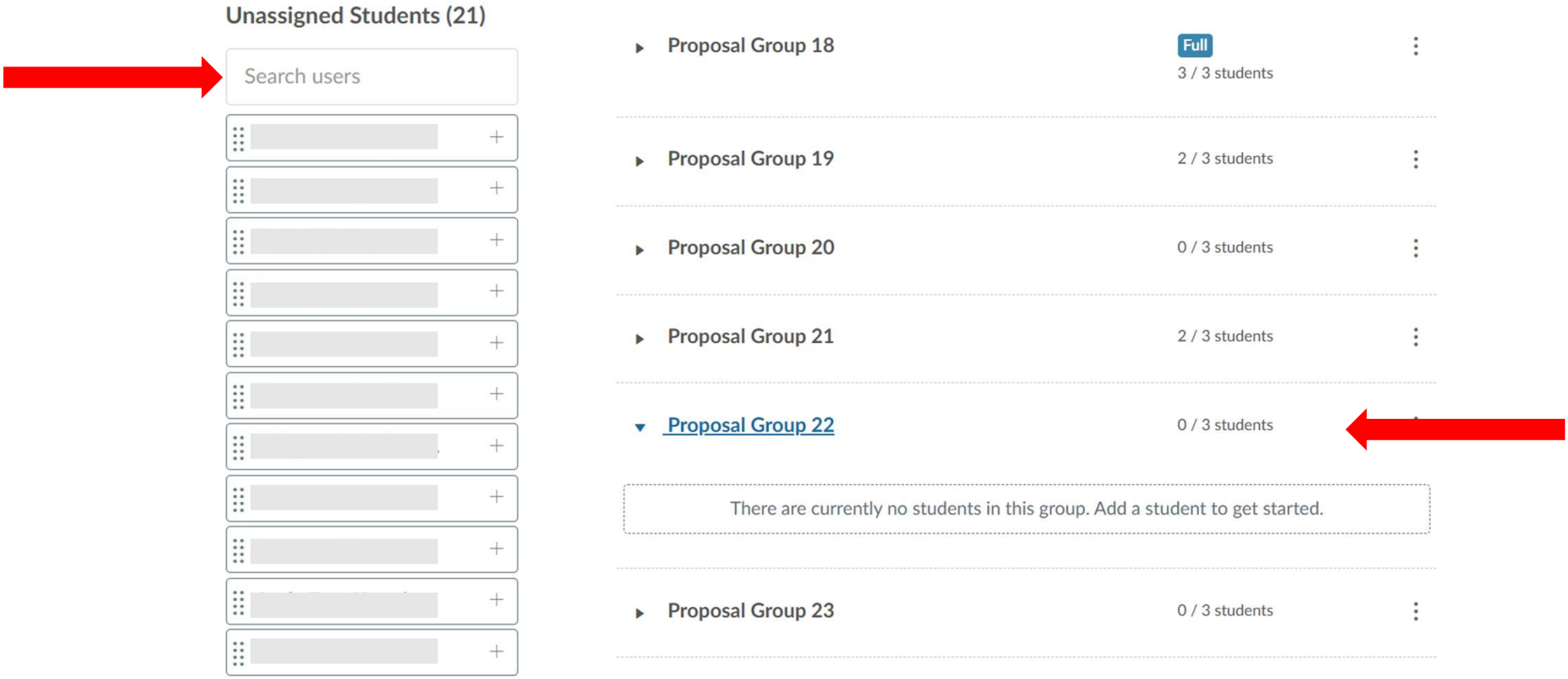
- Project proposals
  - Elevator pitches
  - Mock press releases

- Software development lifecycles (SDLC)
  - What and why are they needed
  - Recurring themes
  - Popular models and their tradeoffs

Canvas project groups for submission



Canvas project groups for submission



Assignment 1 – Project proposals

An **elevator pitch** is a brief, persuasive speech that you use to spark interest in a product, project or idea, or in yourself. An elevator pitch is short, about the time you spend in an elevator, hence the name.

https://asana.com/resources/elevator-pitch-examples

You have 2 minutes for your project pitch to the class - this is a good example of how it could flow

Your turn

Try pitching your project idea, or yourself, to your neighbor

Introduce yourself	
Present the problem	
Present your solution (and technical approach) (This is your lucky day!)	
Share your value proposition (to customer, to devs)	
Add a call to action	

Another tool you'll see used for pitches

Write a **mock product press** release!

Includes

- A catchy headline
- Problem trying to solve
- Value proposition
- How differs from competitors
- Release timing and teaser of future beyond release
- Quotes from well known users showing their delight

Excellent way to paint the vision and get buy in to build it

See: https://jdmeier.com/how-to-create-innovative-disruption-with-mock-press-releases/

https://www.linkedin.com/pulse/working-backwards-press-release-template-example-ian-mcallister

Time to Walk: An inspiring audio walking experience comes to Apple Fitness+

Episodes feature personal stories, photos, and music from influential people to inspire Apple Watch users to walk more

Cupertino, California — Apple today unveiled Time to Walk, an inspiring new audio walking experience on Apple Watch for Fitness+ subscribers, created to encourage users to walk more often and reap the benefits from one of the healthiest activities. Each original Time to Walk episode invites users to immerse themselves in a walk alongside influential and interesting people as they share thoughtful and meaningful stories, photos, and music. Time to Walk can be enjoyed anytime and anywhere with Apple Watch and AirPods or other Bluetooth headphones.

"Walking is the most popular physical activity in the world, and one of the healthiest things we can do for our bodies. A walk can often be more than just exercise: It can help clear the mind, solve a problem, or welcome a new perspective," said Jay Blahnik, Apple's senior director of Fitness Technologies. "Even throughout this challenging period of time, one activity that has remained available to many is walking. With Time to Walk, we're bringing weekly original content to Apple Watch in Fitness+ that includes some of the most diverse, fascinating, and celebrated guests offering inspiration and entertainment to help our users keep moving through the power of walking."

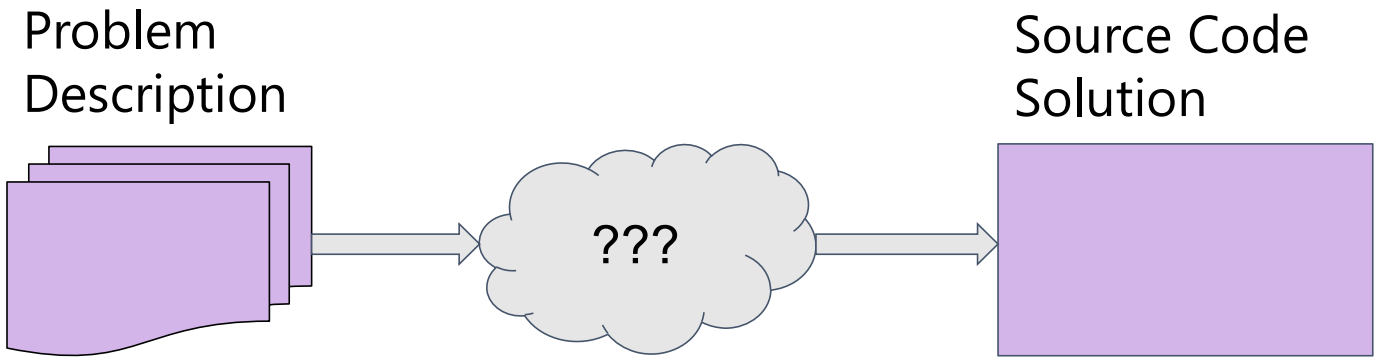
Software Engineering is …

An **engineering discipline** concerned with the complete process of specifying, designing, developing, analyzing and maintaining a software system

Software Engineering tasks include:

- Requirements engineering
- Specification writing and documentation
- Architecture and design
- Programming
- Testing and debugging
- Deploying, operating, evaluating, refactoring and evolving
- Planning, teamwork and communication

Lifecycles: Here's the challenge



One solution: Code and fix

Description or specification (maybe)

Deliver (maybe)

SDLC: Code and fix

Pros:

- Little or no overhead - just dive in and develop, and see progress quickly
- Applicable *sometimes* for small projects, short-lived prototypes, and/or small teams

Cons:

- <Over to you>

SDLC: Code and fix

Pros:

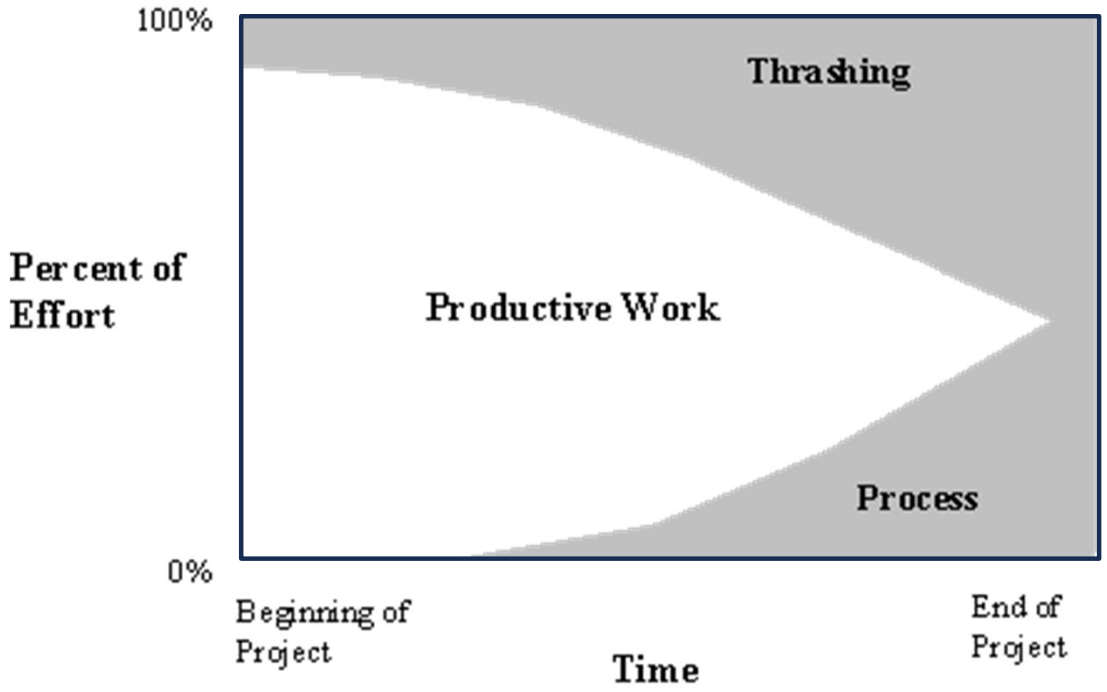
- Little or no overhead - just dive in and develop, and see progress quickly
- Applicable *sometimes* for small projects, short-lived prototypes, and/or small teams

Cons:

- **No way to assess progress, quality or risks**
- **Challenging to manage multiple developers – how synchronize your work**
- Harder to accommodate changes without a major design overhaul
- Unclear delivery of features (scope), timing, and support

Let's look at data

Projects with little attention on SDLC process

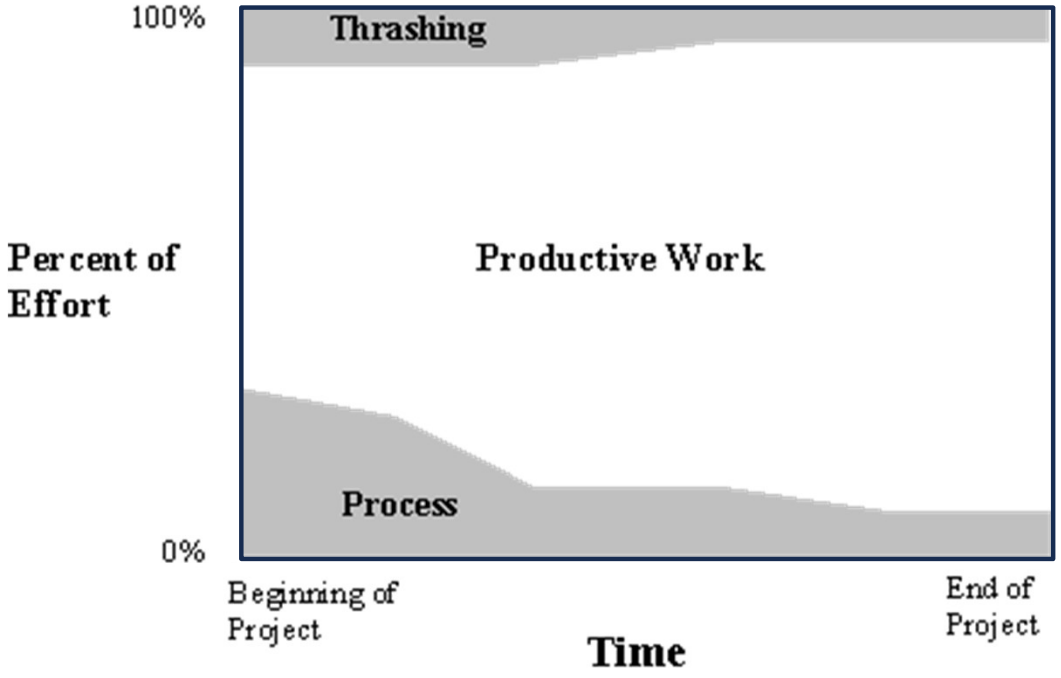


Thrashing: *doing a lot of work but not making progress towards the goal*

[The Power of Process | Steve McConnell]

Let's look at data

Projects with early attention to SDLC process



The Power of Process | Steve McConnell

Is a more structured SDLC necessary?

It's used to establish an order – provide a model - in which software project events occur from project conception to project delivery

- It forces us to think of the "big picture" and follow steps so that we reach it without glaring deficiencies
- Without it we may make decisions that are individually on target but collectively misdirected
- It allows us to organize and coordinate our work as a team
- It allows us to track progress and risks, and adjust as necessary

Recurring themes in SDLCs

A SDLC defines how to produce software through a series of stages

Common stages

- Requirements
- Design
- Implementation
- Testing
- Release
- Maintenance

Goals of each stage

- Define a clear set of actions to perform
- Produce tangible (trackable) items
- Allow for work revision
- Plan actions to perform in the next stage

Key question: how to combine the stages and in what order

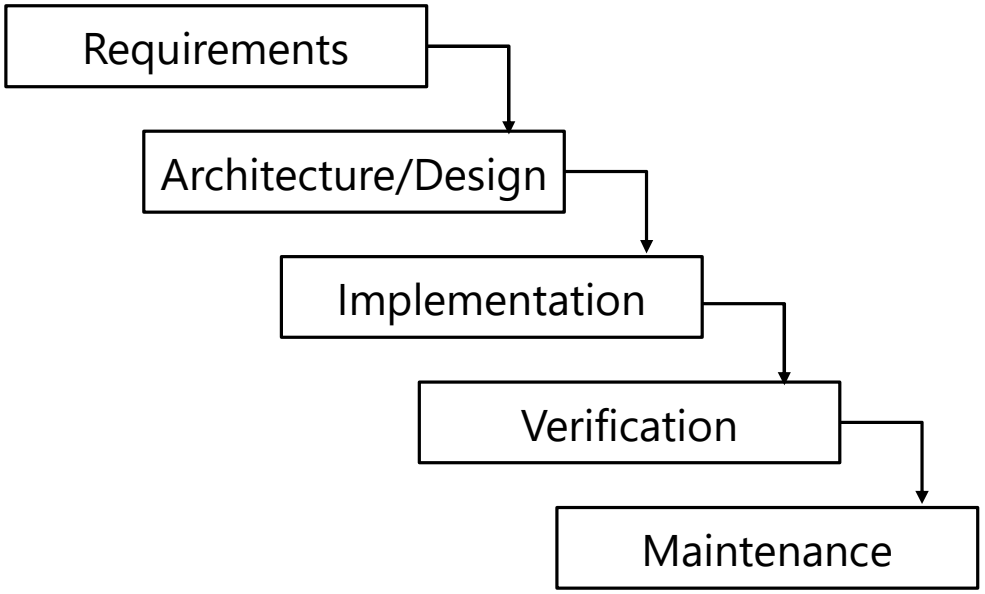
Today's Outline

- Project proposals
  - Elevator pitches
  - Mock press releases

- Software development lifecycles (SDLC)
  - What and why are they needed
  - Recurring themes
  - **Popular models and their tradeoffs**
    - Waterfall model
    - Prototyping
    - Spiral model
    - Staged delivery
    - Agile (XP, Scrum)

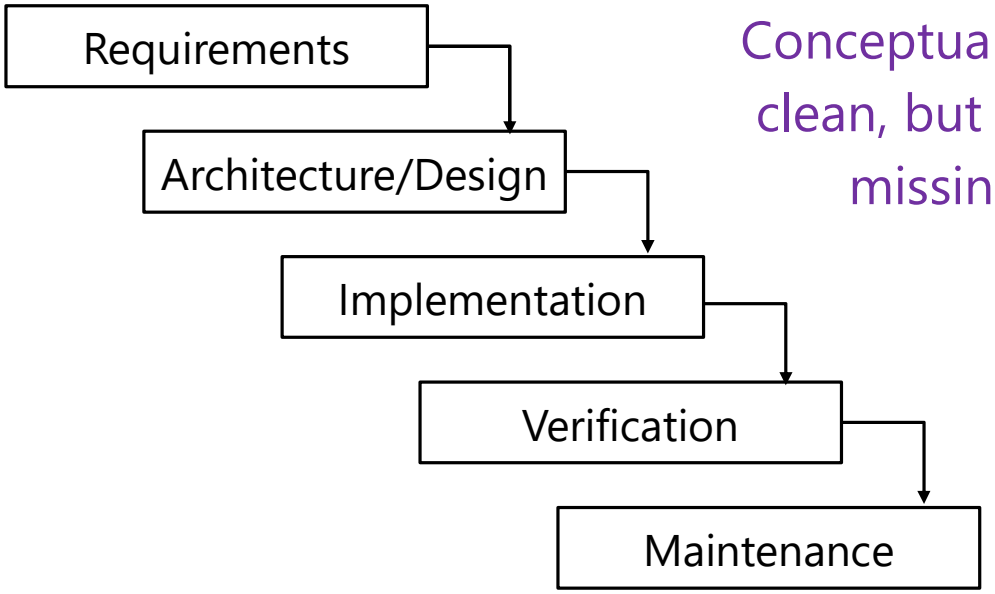
All have the same goal – deliver high quality software, on time, meeting the customers needs

SDLC: Waterfall model



- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document

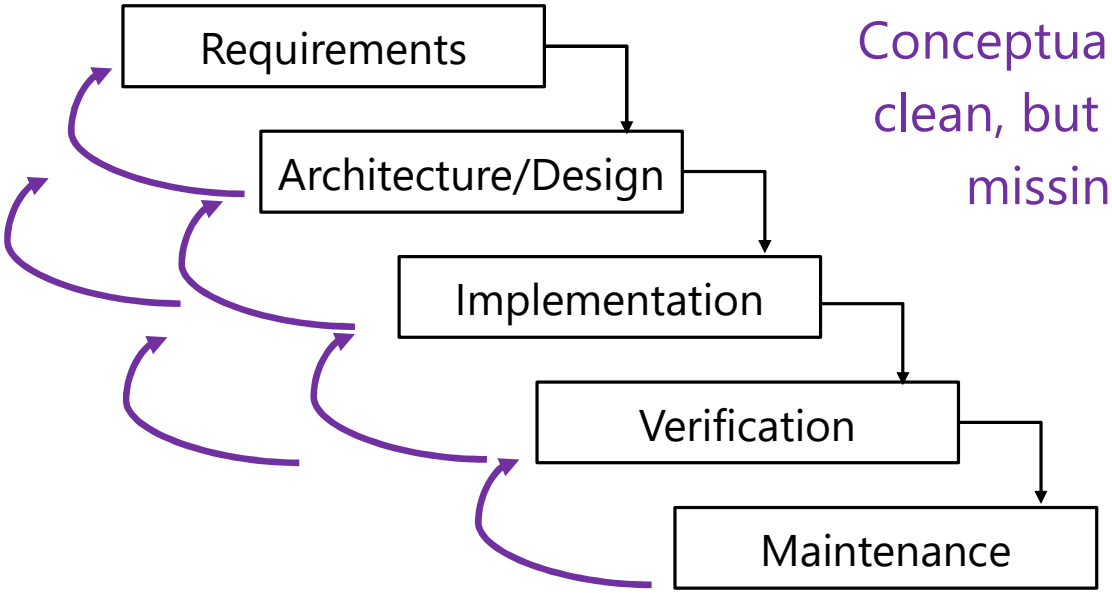
SDLC: Waterfall model



Conceptually very clean, but what's missing?

- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document

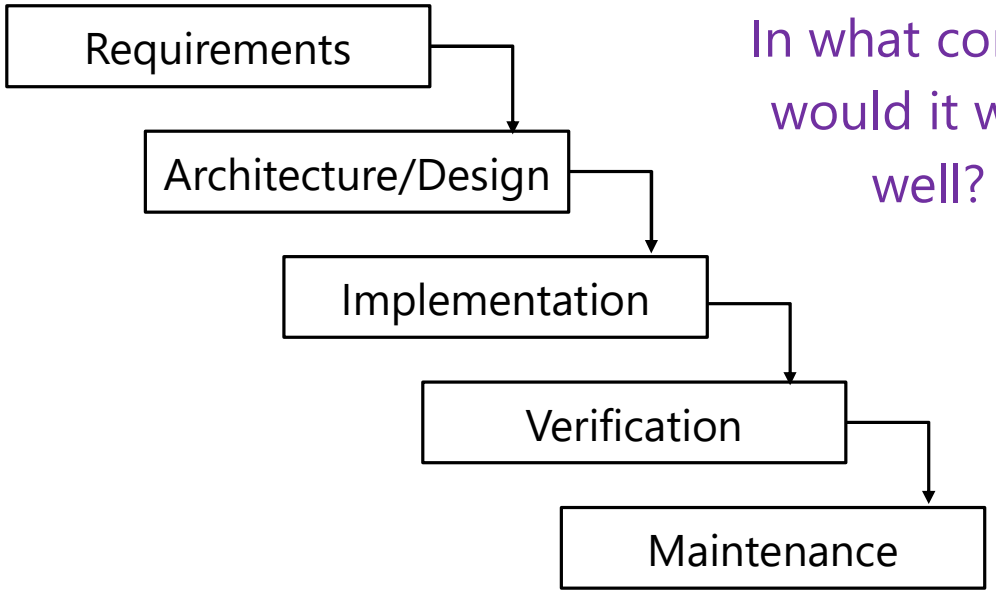
SDLC: Waterfall model



Conceptually very clean, but what's missing?

- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document

SDLC: Waterfall model



In what context would it work well?

- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document

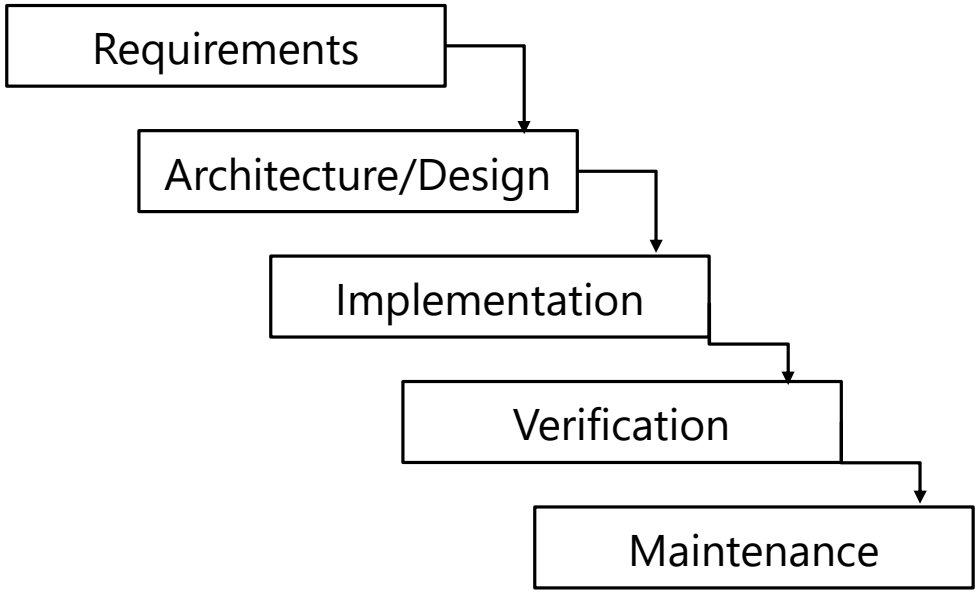
Honeywell's Flight Management System Selected By Airbus

Honeywell's solution will address the avionics needs of the Airbus A320, A330 and A350 aircraft fleet

Ahjay Rai
May 19, 2022

Likely parts of their SDLC is waterfall-like due to the upfront and regulated requirements

SDLC: Waterfall pros and cons



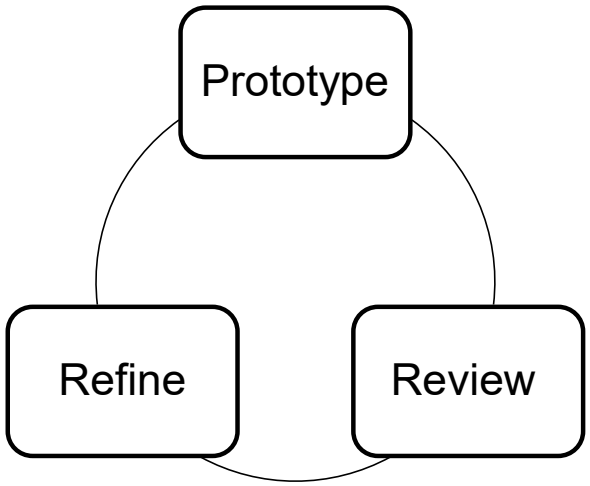
Pros:

- Simple to understand
- Promotes common dialogue
- Highly regulated deliverables

Cons:

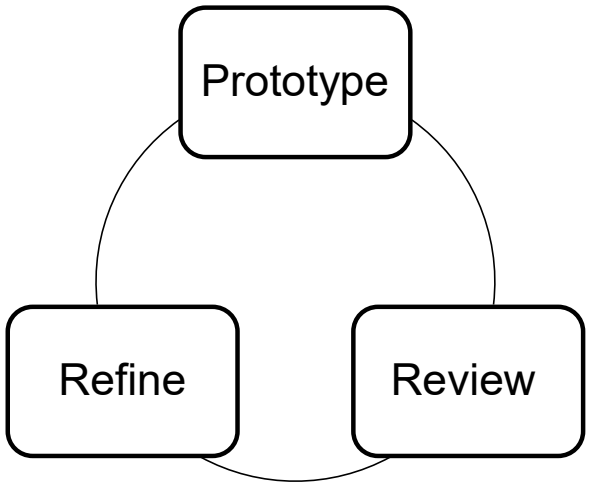
- Hard to do all the planning upfront
- Inflexible – changes are expensive
- Test and integration come late – fixes are expensive
- Final product may not match the customer's needs

SDLC: [Rapid | Evolutionary] Prototyping



- Problem domain or requirements not well defined or understood
- Create small implementations of requirements that are least understood
- Requirements are "explored" before the product is fully developed
- Developers (and customers) gain experience when developing the product
- Prototype can evolve to the real product *or* can serve to be a learning tool only

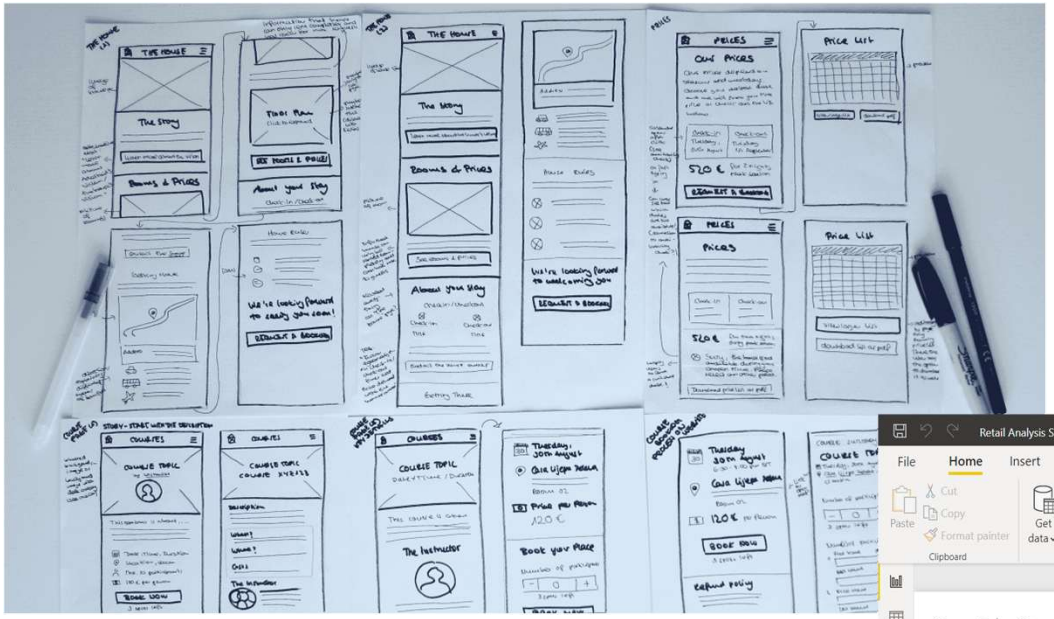
SDLC: Prototyping



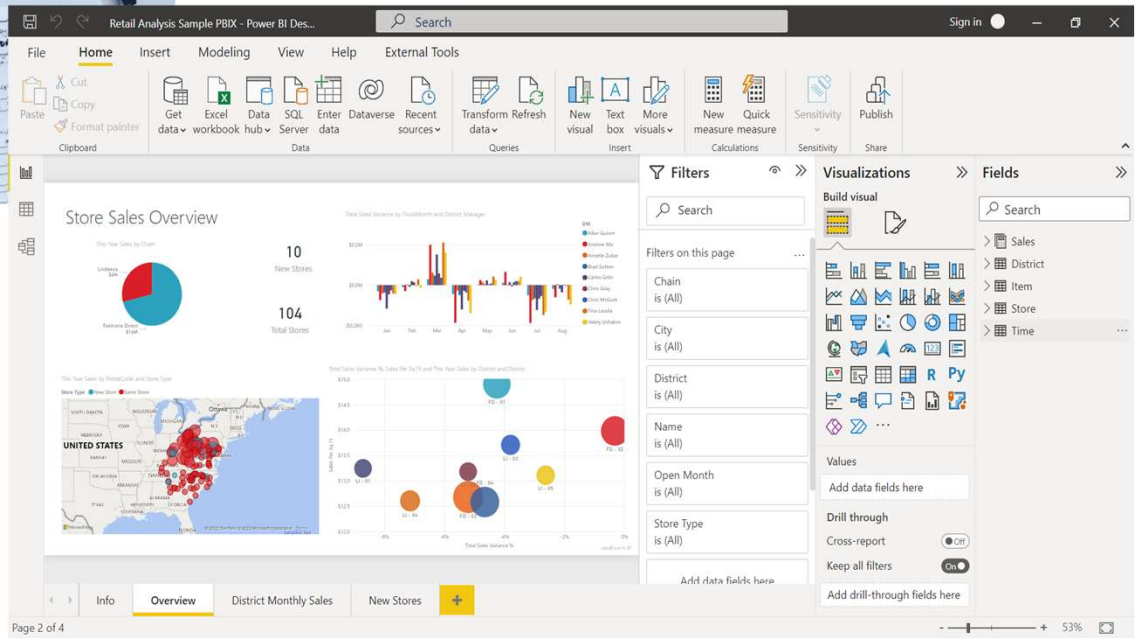
In what context would it work well?

- Problem domain or requirements not well defined or understood
- Create small implementations of requirements that are least understood
- Requirements are "explored" before the product is fully developed
- Developers (and customers) gain experience when developing the product
- Prototype can evolve to the real product *or* can serve to be a learning tool only

UI prototyping is popular

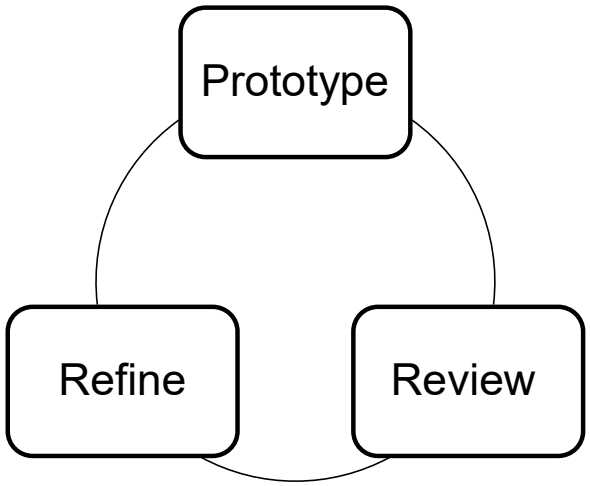


https://internetdevels.com/blog/what-is-website-prototype-how-build-website-prototype



https://learn.microsoft.com/en-us/power-bi/fundamentals/desktop-what-is-desktop

SDLC: Prototyping pros and cons



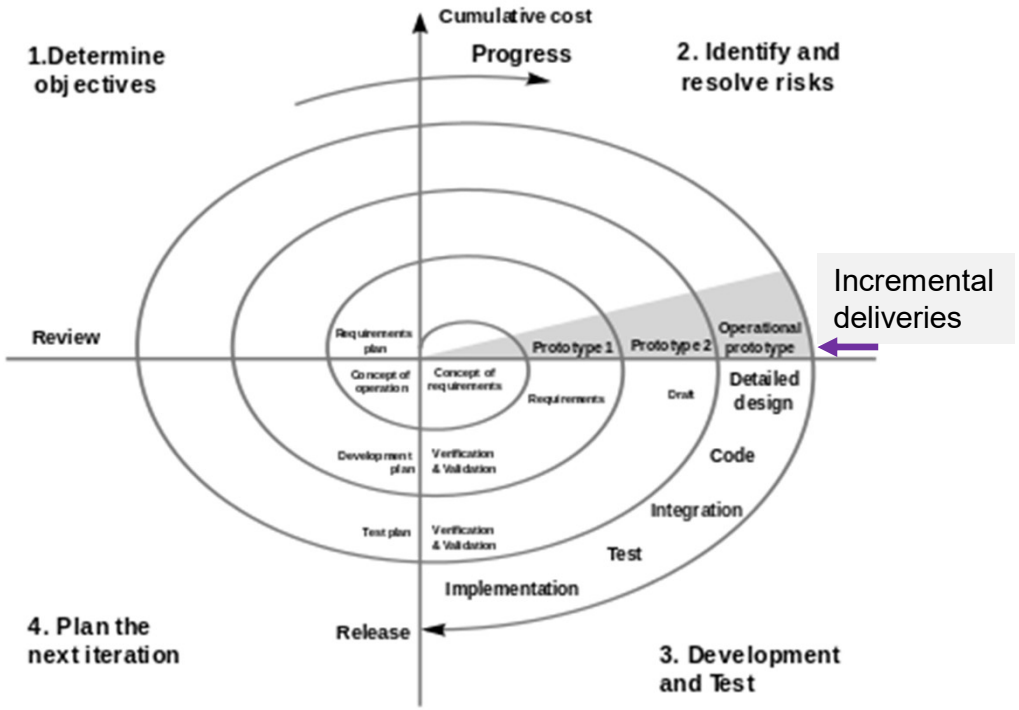
Pros:

- Client involvement and early feedback
- Improves requirements and specifications
- Reduces risk of developing the "wrong" product

Cons:

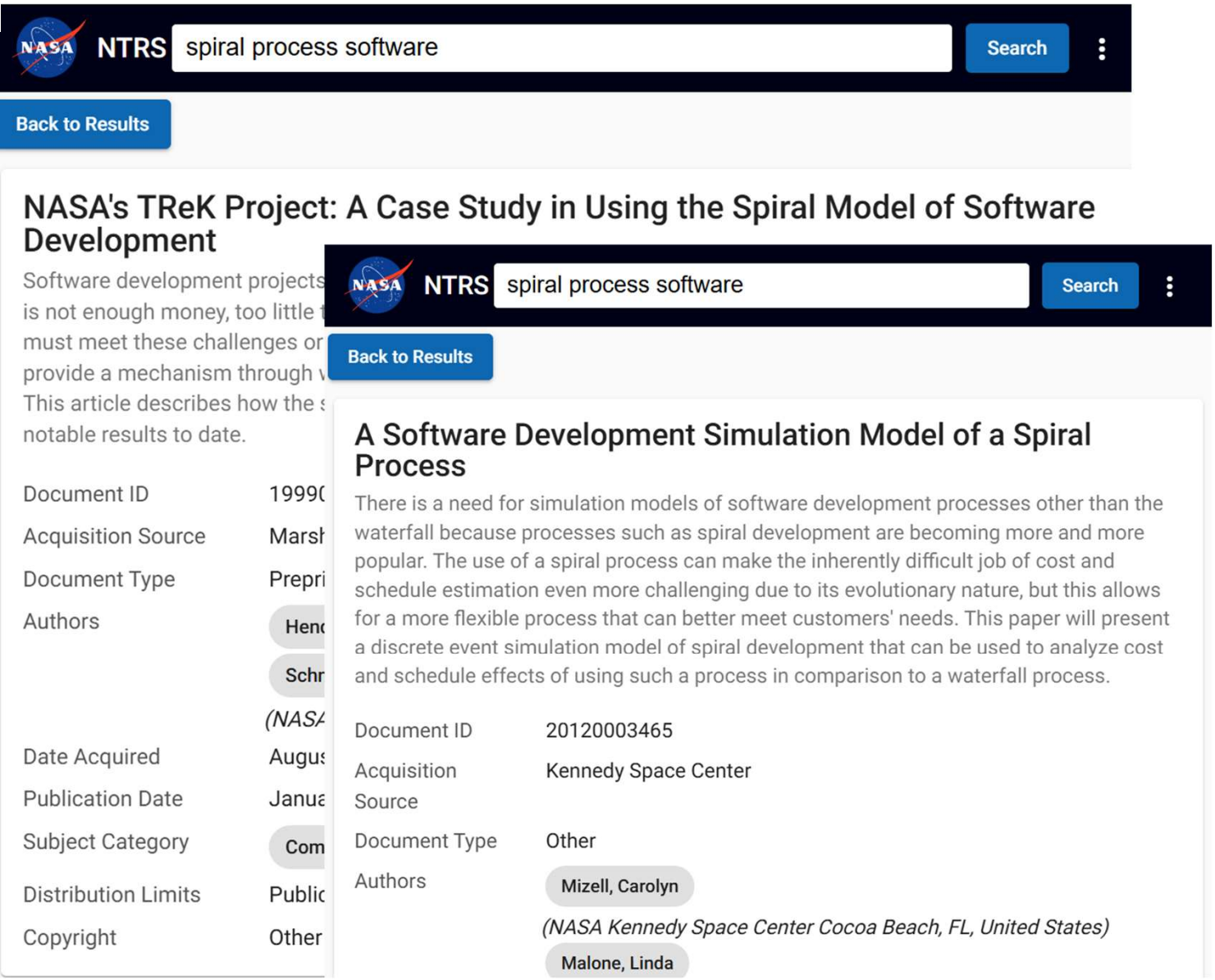
- Time/cost for developing may be high
- Hard to commit what will be delivered and when
- May end up evolving a poor choice (limit thinking holistically)

SDLC: Spiral Model



- Incremental/iterative model (combines waterfall and prototyping)
- Iterations called spirals
- Repeat these activities:
  1. Determine objectives (reqs)
  2. Risk analysis
  3. Develop and test
  4. Plan next delivery
- Phased reduction of risks (address high risks early)

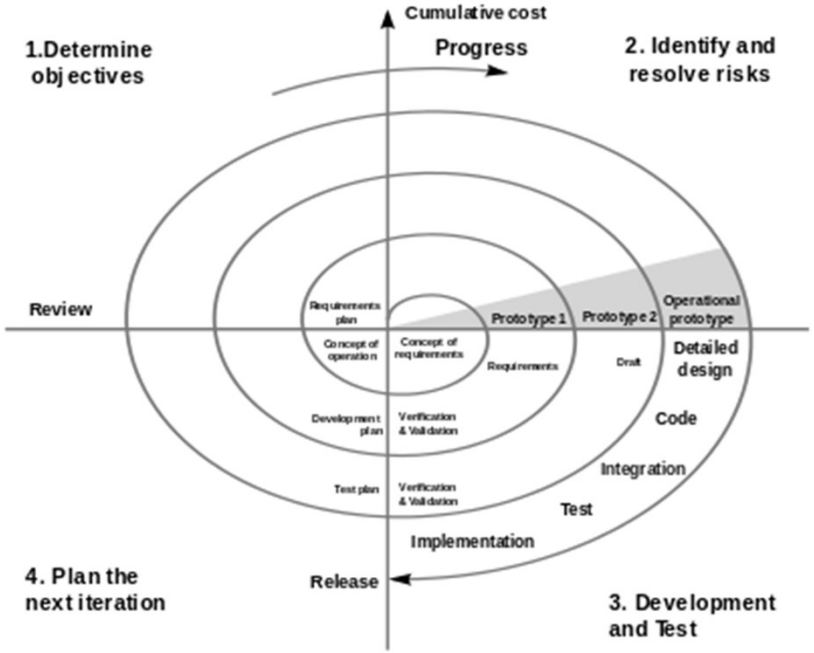
Boehm, *Spiral Development: Experience, Principles,and Refinements*



Key Aspects and Pros of Spiral Approach

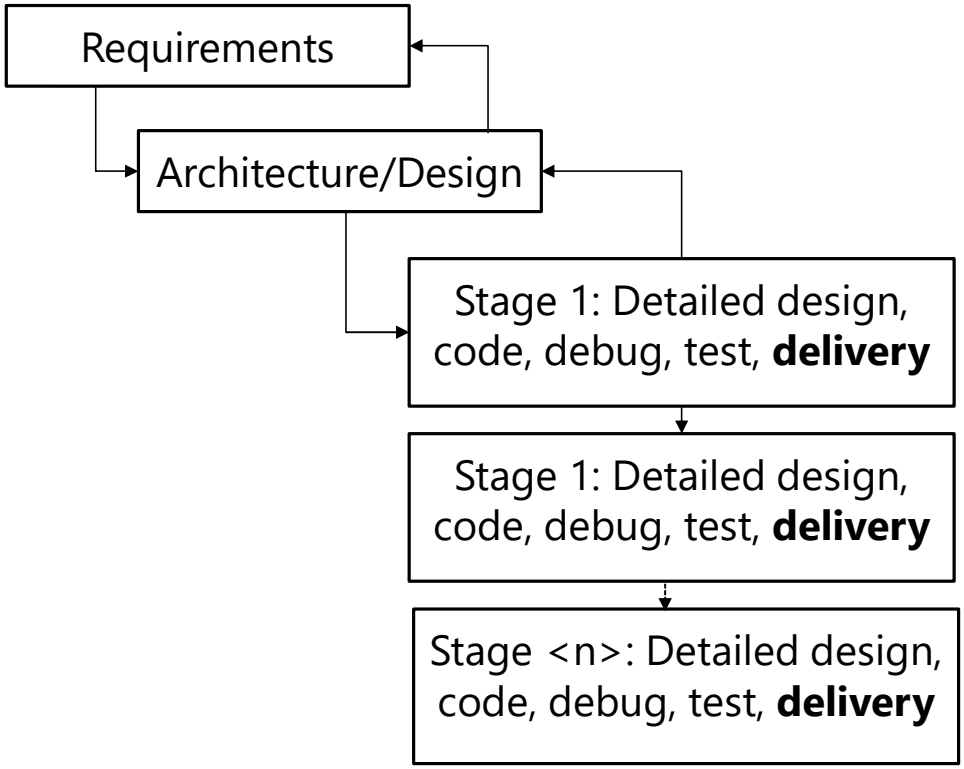
- Risk-Driven: Identify and mitigate risks early and continuously
- Flexible & Incremental: Evolve the project in cycles, with each loop adding functionality or improving existing systems
- Stakeholder Involvement: All parties participate in each cycle to ensure all needs are addressed and consensus reached

SDLC: Spiral Model importance



- Interesting to us as it's a **precursor to agile models**
- Software development is based on **iteration**, using "**risk reduction**" as the criteria to prioritize activities at each iteration

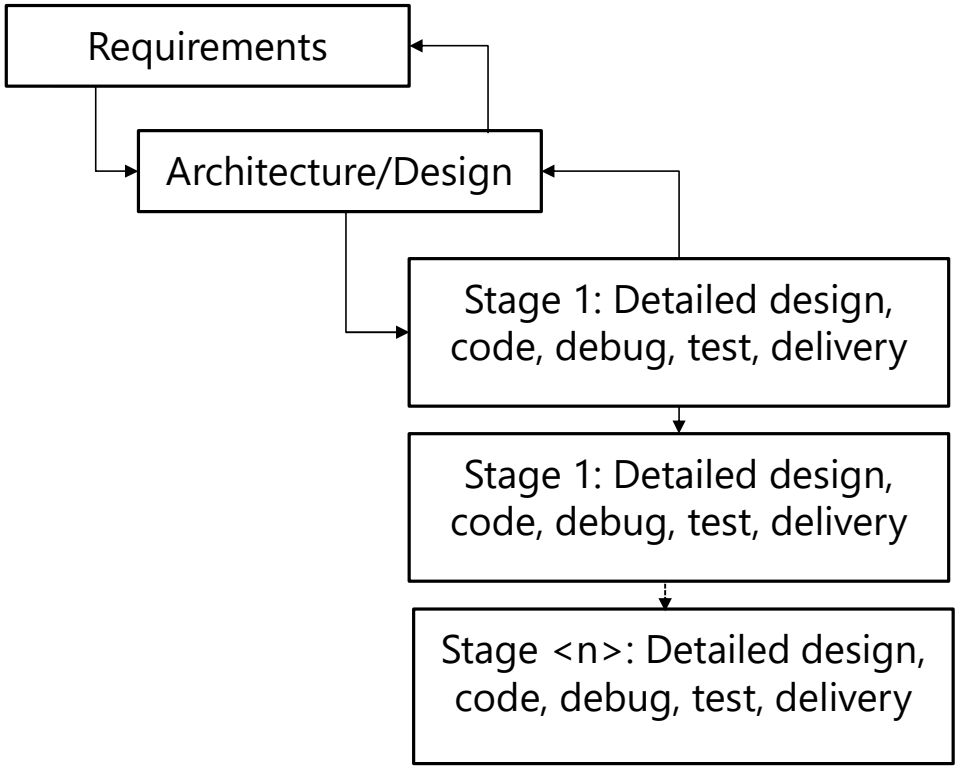
SDLC: Lots of variants 🤯 - Staged Delivery



- Waterfall-like planning upfront then spiral/scrum-like short release cycles
- Pros: ?
- Cons: ?

McConnell: https://stevemcconnell.com/

SDLC: Staged Delivery pros and cons



- Pros:
  - Can ship at the end of any release cycle
  - Intermediate deliveries show progress, satisfy customers, and lead to feedback
  - Problems are visible early
- Cons:
  - Requires tight coordination
  - Product must be decomposable
  - Extra releases cause overhead

Today's Outline

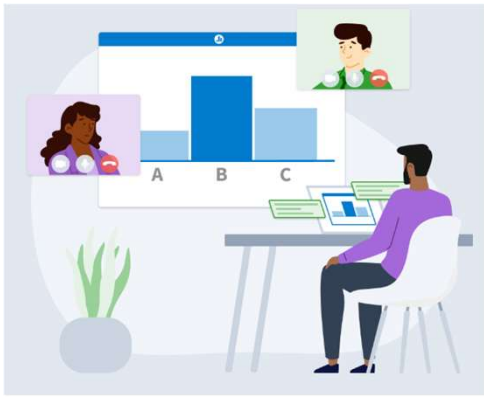
- Project proposals
  - Elevator pitches
  - Mock press releases

- Software development lifecycles (SDLC)
  - What and why are they needed
  - Recurring themes
  - **Popular models and their tradeoffs**
    - Waterfall model
    - Prototyping
    - Spiral model
    - Staged delivery
    - **Agile (XP, Scrum)**

Traditional models

Pop Quiz – true or false

https: //PollEv.com/cse403wi



1. Waterfall is the **best** SDLC for a brand new online rock-climbing game

2. Staged delivery is the **best** SDLC for a contract to provide drone-delivery software by December 2026

3. Evolutionary prototyping is the **best** SDLC to use for a driver-less car navigation system

Best traditional SDLC: true or false

0 surveys completed

0 surveys underway

Waterfall is the best SDLC for a brand new online rock-climbing game

True

False

Staged delivery is the best SDLC for a contract to provide drone-delivery software by Dec 2026

True

False

Evolutionary prototyping is the best SDLC to use for a driver-less car navigation system

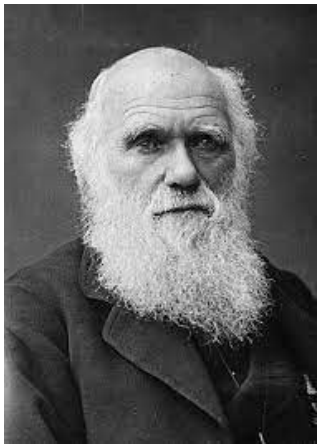
True

False

Onto Agile models

What is Agile all about?

Premise: the world is too uncertain, and we must be flexible and responsive to changes



There is nothing permanent except change - Heraclitus (Greek philosopher)

It is not the strongest or the most intelligent who will survive but those who can best manage change - Charles Darwin (English naturalist)

Agile Manifesto

A Behind the Scenes Look at the Writing of the Agile Manifesto

Agile Manifesto (http://agilemanifesto.org/):

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

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

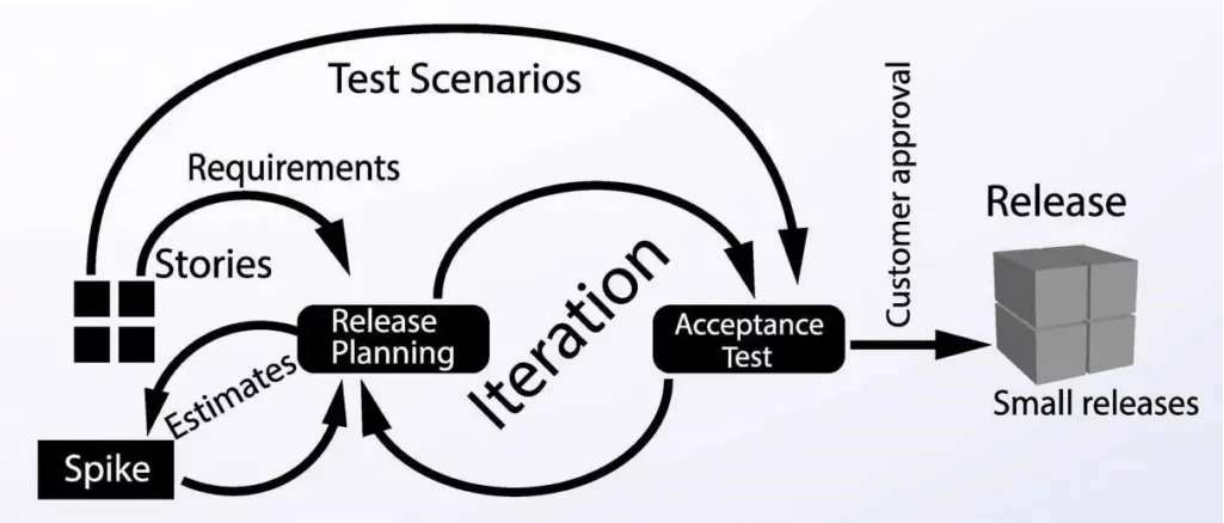
Agile models

"Agile software development" is a general term for values, frameworks and practices outlined in the Agile Manifesto

Agile models

- Aim to deliver a high-quality product to the customer as fast as possible
- Focus on simplicity, excellence, continuous testing, integration
- Incremental and frequent delivery of working software
- Continuous customer involvement
- Expect requirements to change

Agile SDLC: Extreme Programming (XP)



https://www.nimblework.com/agile/extreme-programming-xp/

- XP emphasizes how engineers should work – <u>good practices taken to an extreme</u>
- Examples:
  - Continuous testing and integration
  - 10-minute build
  - Constant discussions with customers
  - Full flexibility to change requirements anytime
  - Pair programming
  - Test-driven development

Agile and XP

12 Agile Manifesto Points

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together** daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. **Working software is the primary measure of progress**.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to **maintain a constant pace indefinitely**.
9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity**—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the team reflects on how to become more effective, then tunes and **adjusts its behavior** accordingly.

12 XP Practices

Fine-scale feedback
- **Pair programming**
- Planning game
- **Test-driven development**
- Whole team

Continuous process
- Continuous integration
- Refactoring or design improvement
- Small releases

Shared understanding
- Coding standards
- Collective code ownership
- Simple design
- System metaphor

Programmer welfare
- Sustainable pace

XP Practice: Pair Programming

Pair programming – All production software is developed by two people sitting at the same machine

Provides for continuous code development, collaboration and review

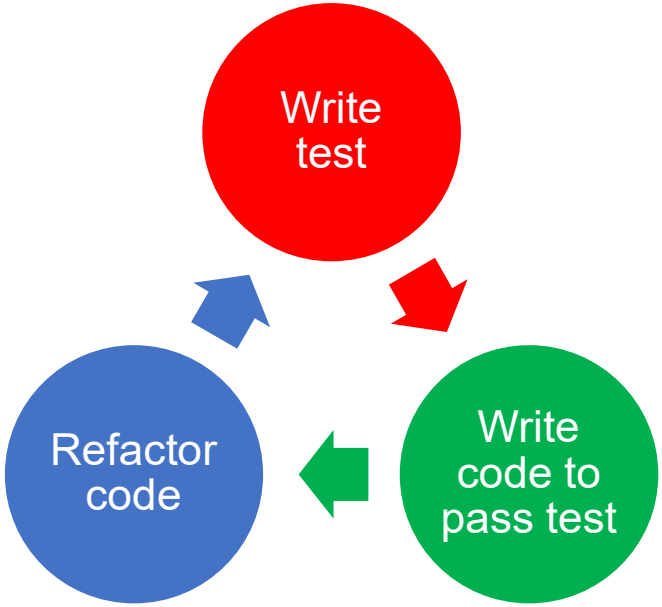
Thoughts?

XP Practice: Test driven development

Write tests based on the requirements - before the production code is even written - and then develop code to make the tests pass

Tests run early and often

Thoughts?



Why are there so many SDLC models?!

Choices are good ☺!

- **The choice depends on the project context and requirements**
- All models have the same goals: manage risks and produce high quality software
- **All models involve the same general activities and stages** (e.g., specification, design, implementation, and testing) and can be tailored
- Today's models involve **customer** feedback and the ability to adapt to **changing requirements**