

Software Development Lifecycles

CSE 403 Software Engineering

Today's Outline

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

CSE Affiliate Professor and 403 Instructor

- Gail Alverson, Ph.D.



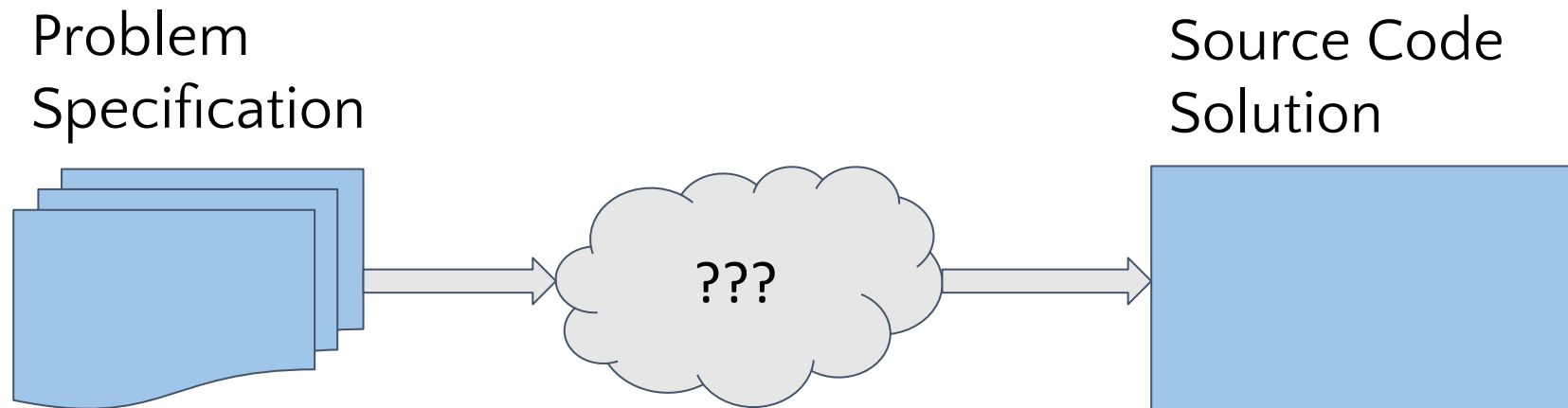
Software Engineering is ...

“An **engineering discipline** concerned with all aspects of **software production** from the early stages of system specification [requirements] through to maintaining [evolving] the system after it has gone into use.” – Ian Sommerville

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

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:

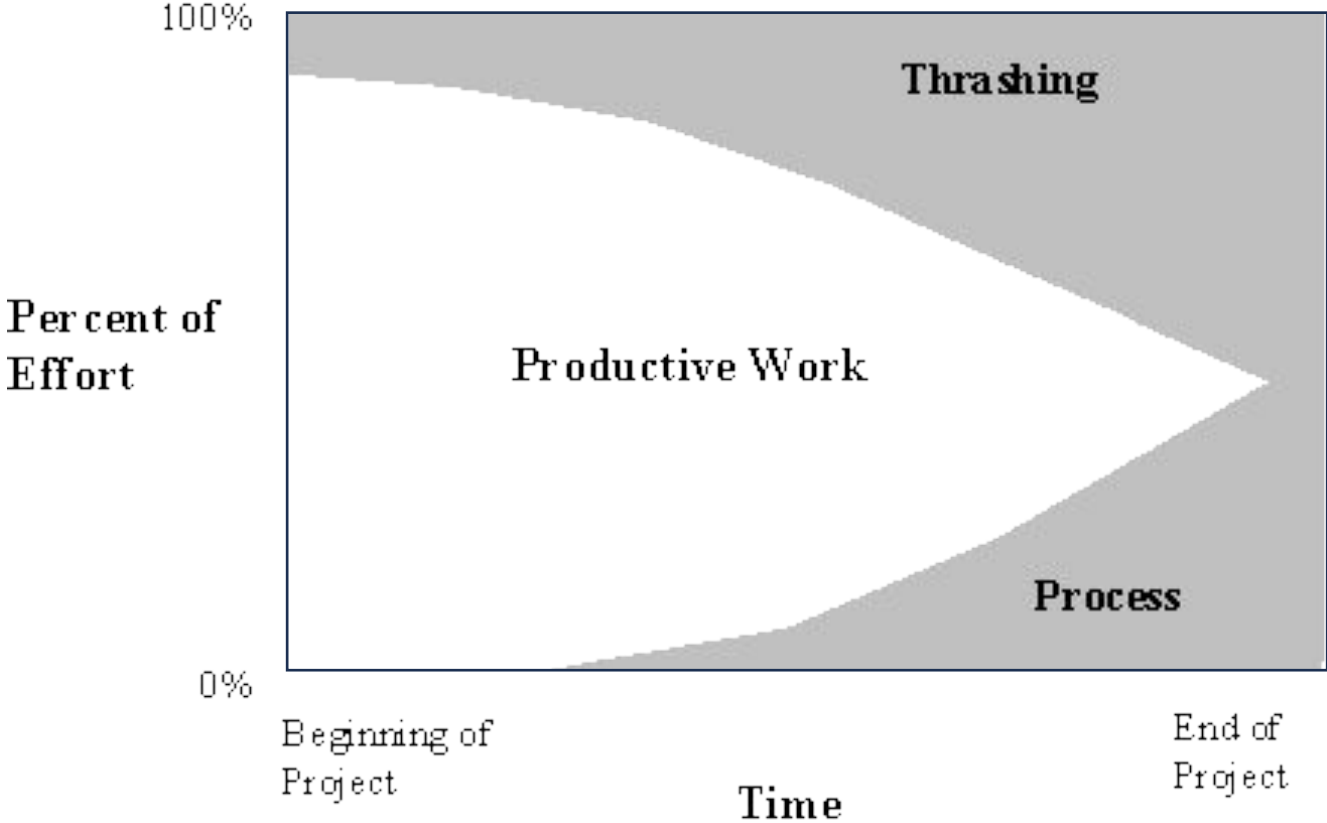
- Little or no overhead – just dive in and develop, and see progress quickly
- Applicable *sometimes* for small projects, short-lived prototypes, and/or small teams

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

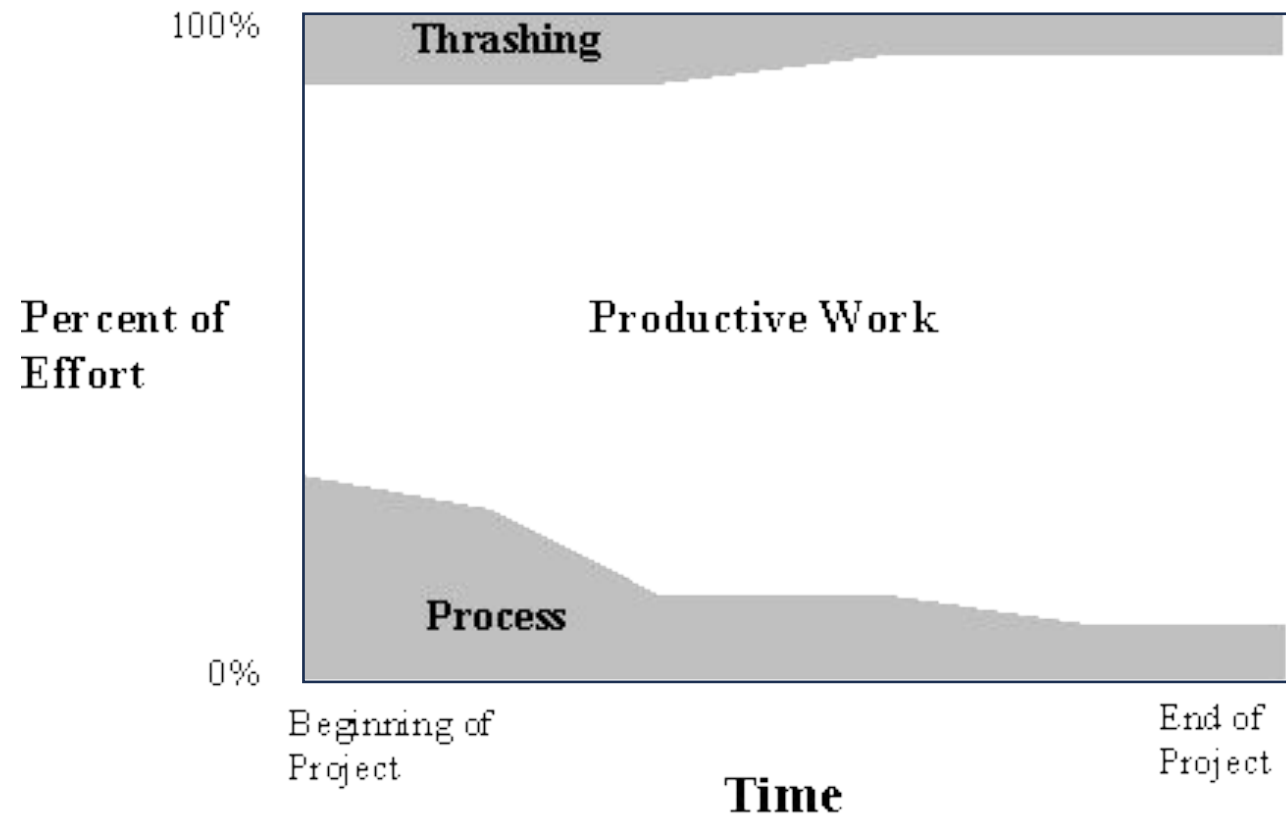
Project with little attention on SDLC process



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

Let's look at data

Project with early attention to SDLC process



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

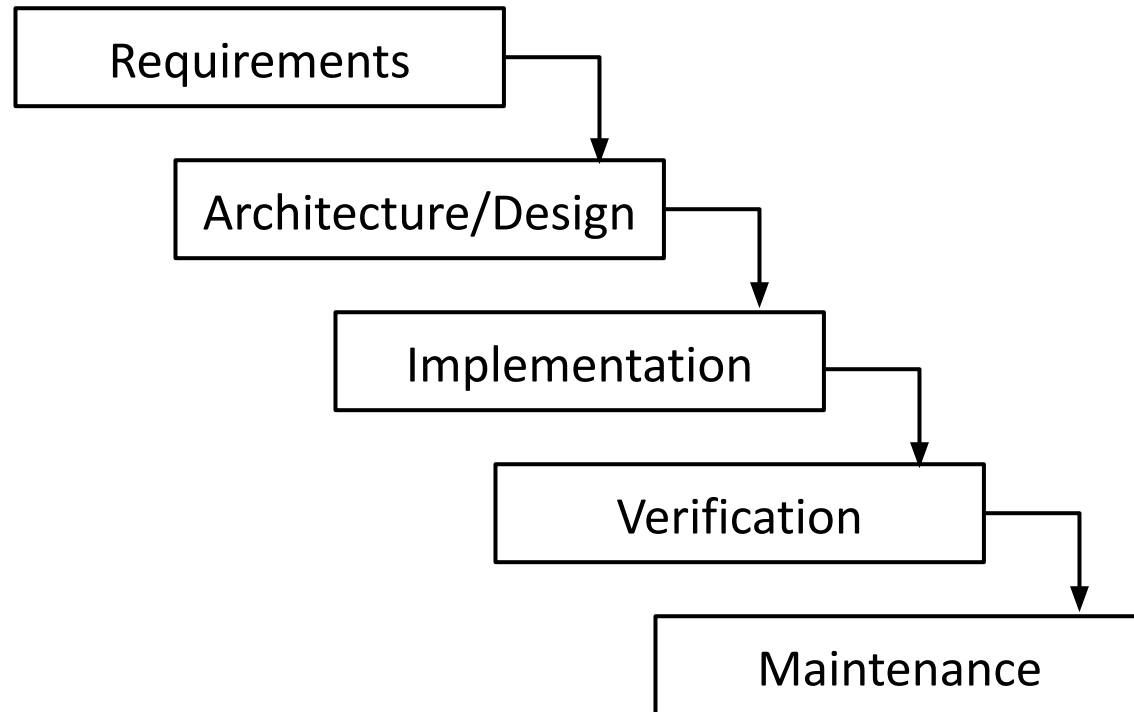
Today's Outline

- Quick introduction
- 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)



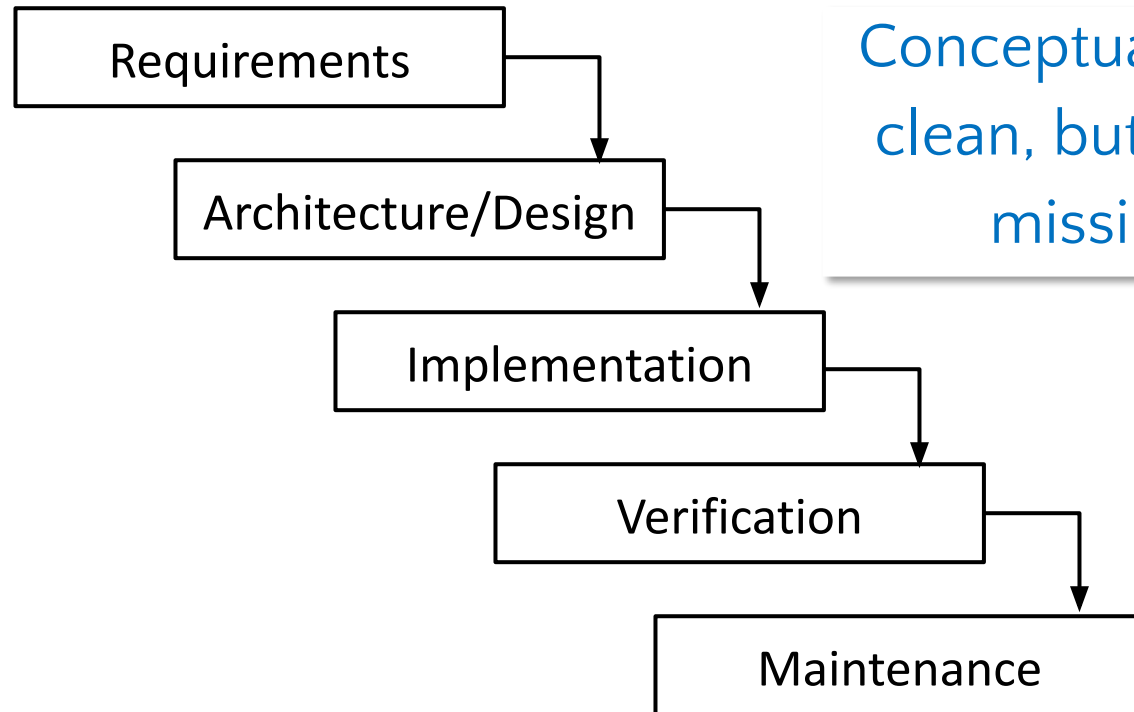
All have the same goal – deliver high quality software, on time, meeting the customers needs

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

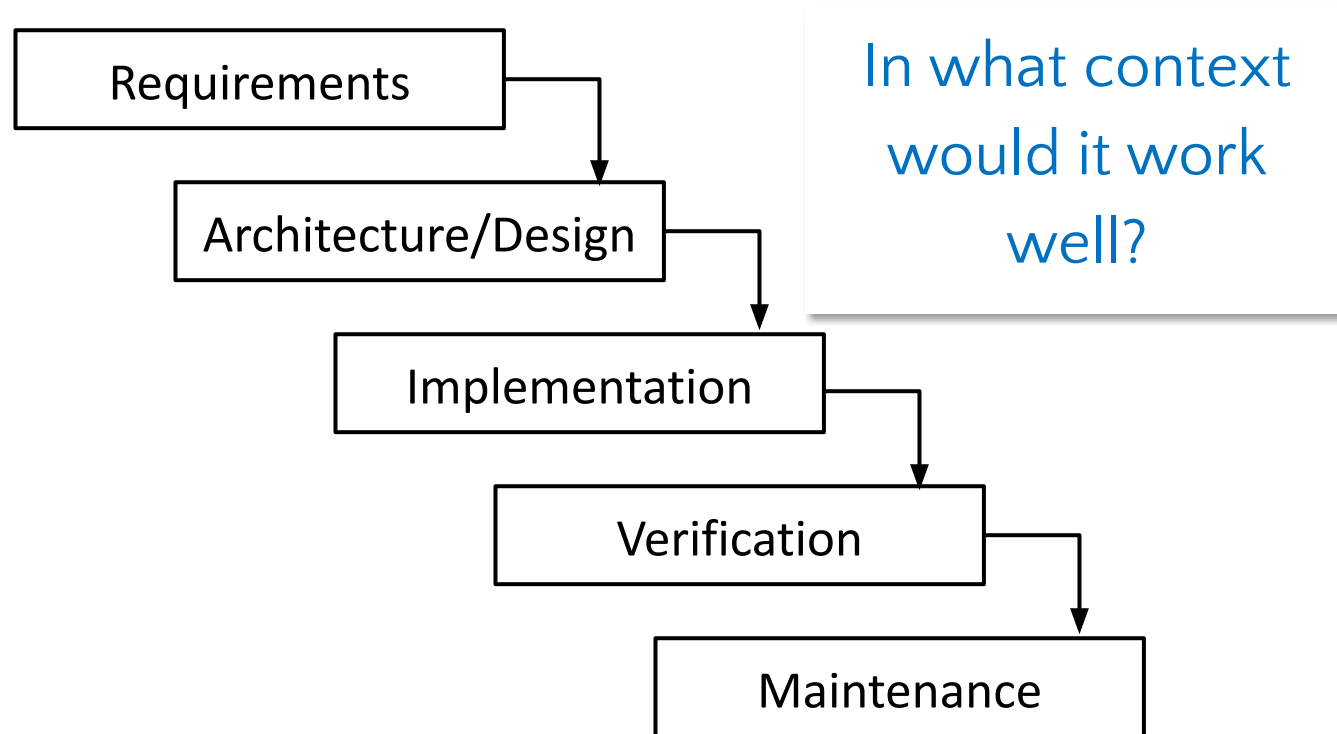
SDLC: Waterfall model



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



- 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

A screenshot of the FDA website. The header includes the FDA logo and 'U.S. FOOD & DRUG ADMINISTRATION'. The breadcrumb trail shows: Home / Medical Devices / Device Advice: Comprehensive Regulatory Assistance / Overview of Device Regulation. The main heading is 'Overview of Device Regulation'. Below it are social sharing icons for Facebook, Twitter, LinkedIn, Email, and Print. The page content includes a sub-heading 'Introduction' and a paragraph: 'FDA's Center for Devices and Radiological Health (CDRH) is responsible for regulating firms who manufacture, repackage, relabel, and/or import medical products in the United States. In addition, CDRH regulates radiation-emitting products (medical and non-medical) such as lasers, x-ray equipment, microwave ovens and color televisions.' A sidebar on the right contains a 'Contents' section with a list of topics including 'of:', '09/04', 'Regu', 'Prod', 'Medi', 'Radia', and 'Prod'.

FDA U.S. FOOD & DRUG ADMINISTRATION

← Home / Medical Devices / Device Advice: Comprehensive Regulatory Assistance / Overview of Device Regulation

Overview of Device Regulation

Share Tweet LinkedIn Email Print

Overview of Device Regulation

A History of

Introduction

FDA's Center for Devices and Radiological Health (CDRH) is responsible for regulating firms who manufacture, repackage, relabel, and/or import medical products in the United States. In addition, CDRH regulates radiation-emitting products (medical and non-medical) such as lasers, x-ray equipment, microwave ovens and color televisions.

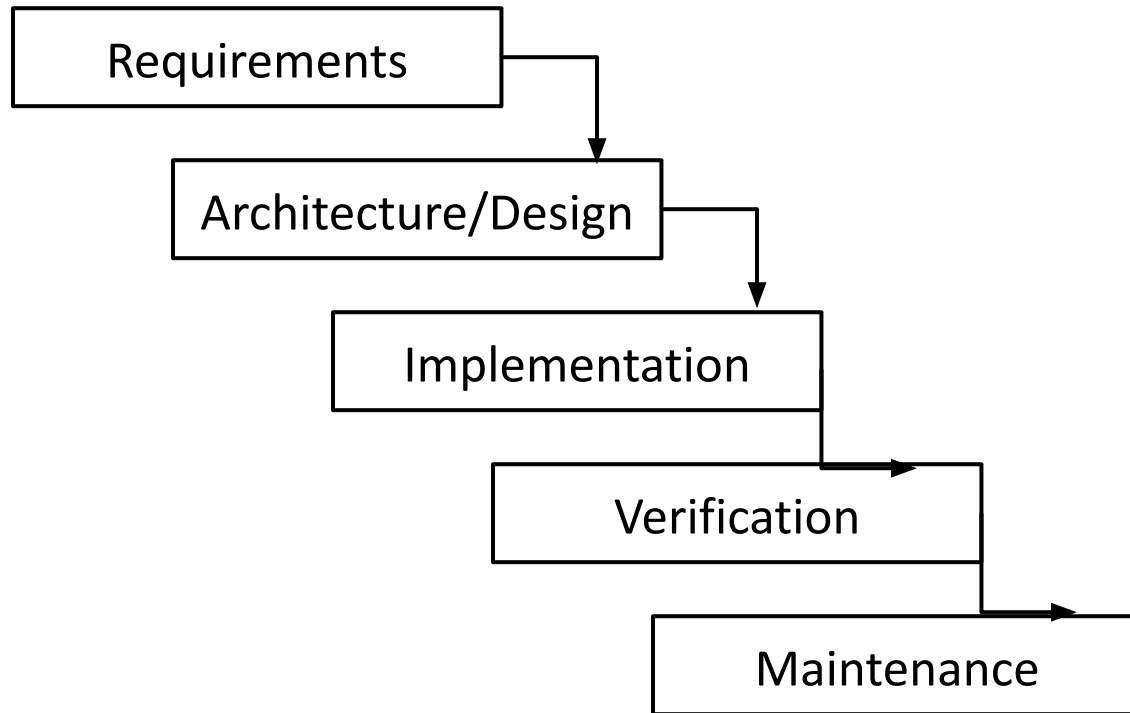
Electronic Products

Contents

- of:
- 09/04
- Regu
- Prod
- Medi
- Radia
- Prod



SDLC: Waterfall pros and cons



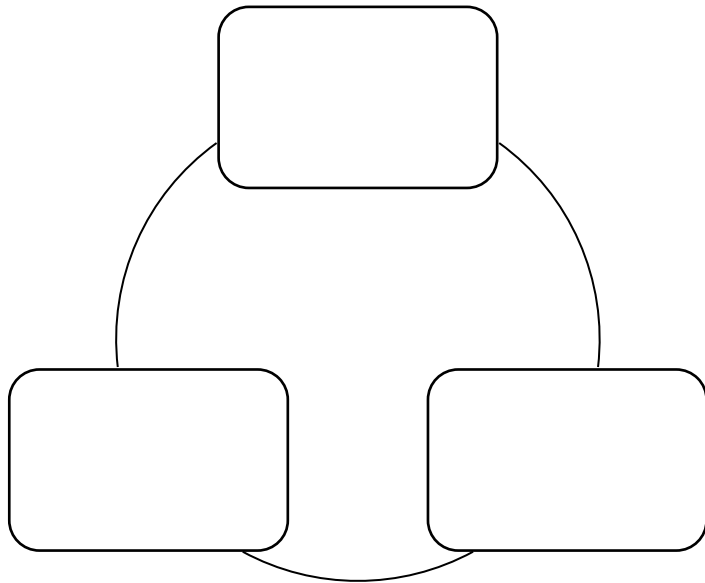
Pros:

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

Cons:

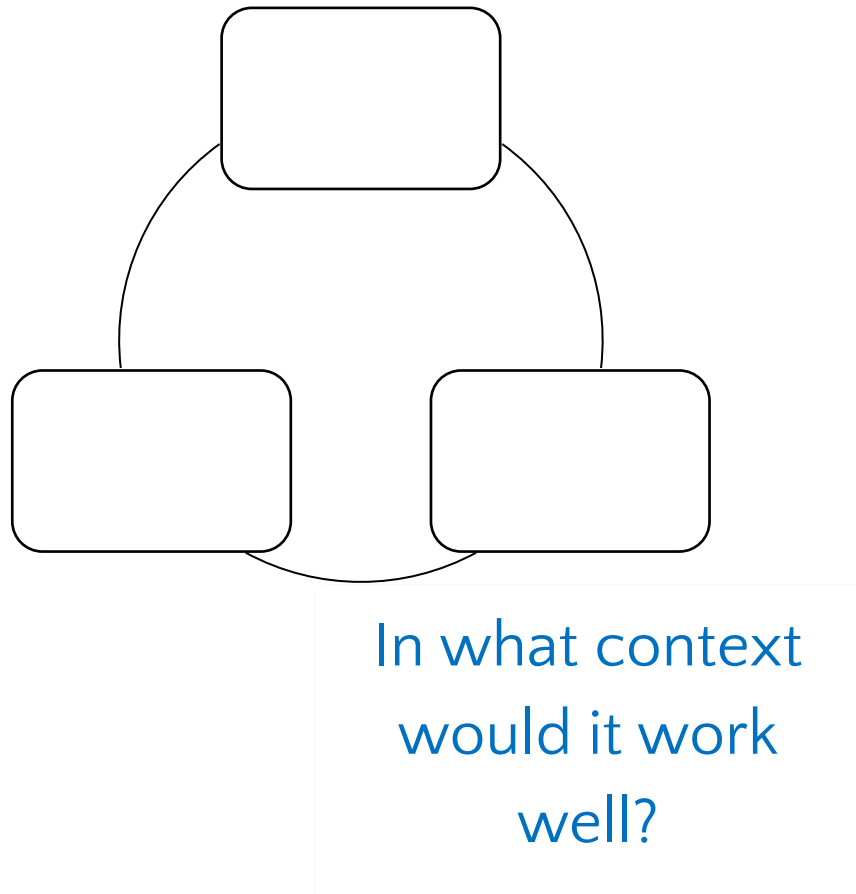
- Hard to do all the planning upfront
- Inflexible – changes are expensive
- Test and integration come late – fixes are expensive
- Final product may not match the customer's needs

SDLC: Prototyping

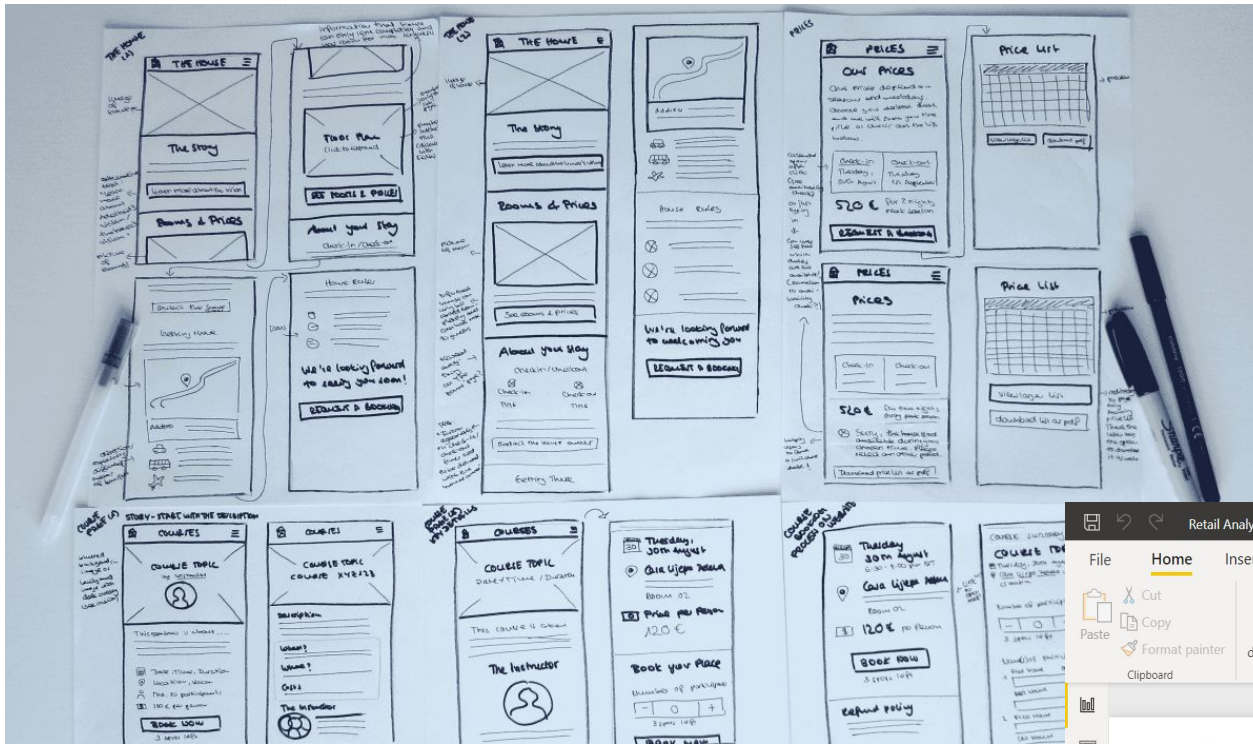


- Problem domain or requirements not well defined or understood
- Create small implementations of requirements that are least understood
- Requirements are “explored” before the product is fully developed
- Developers (and customers) gain experience when developing the product
- Prototype can evolve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

SDLC: Prototyping

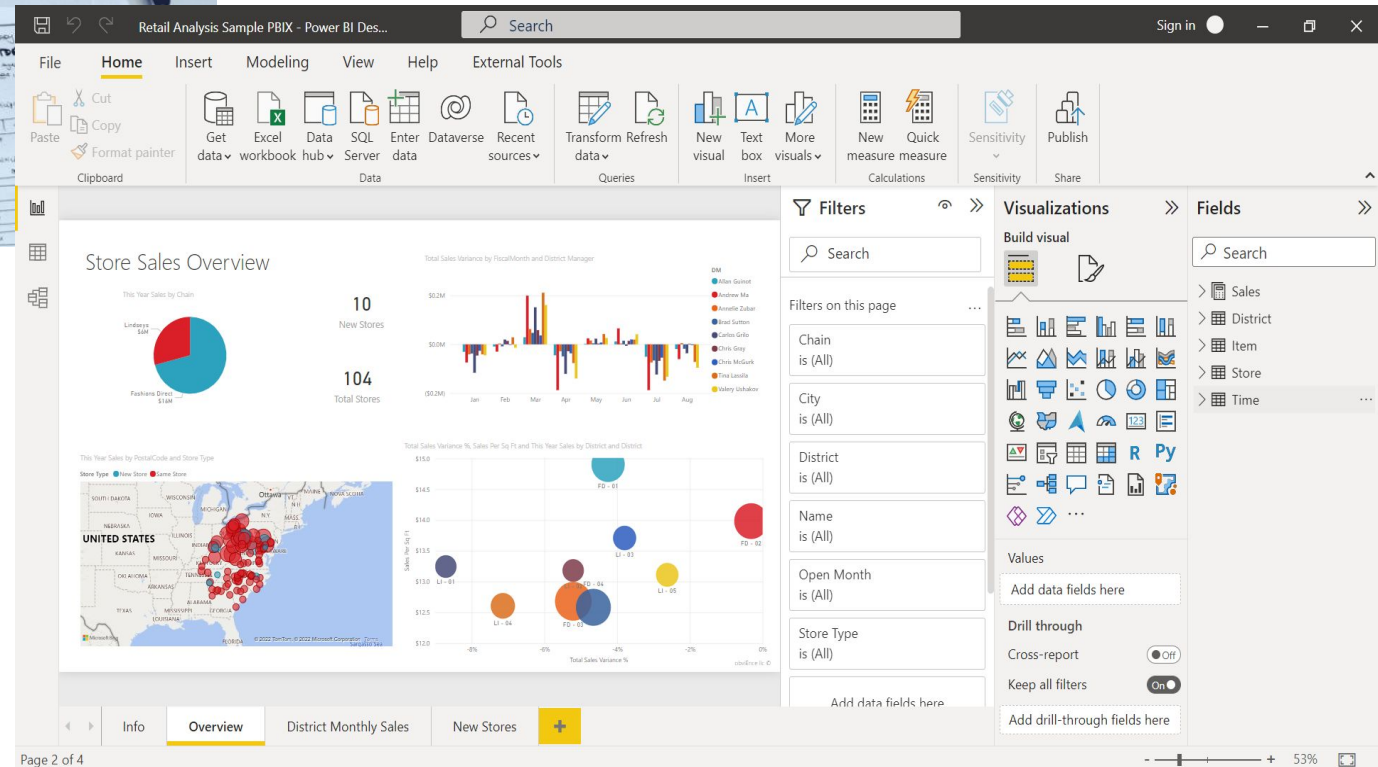


- Problem domain or requirements not well defined or understood
- Create small implementations of requirements that are least understood
- Requirements are “explored” before the product is fully developed
- Developers (and customers) gain experience when developing the product
- Prototype can evolve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



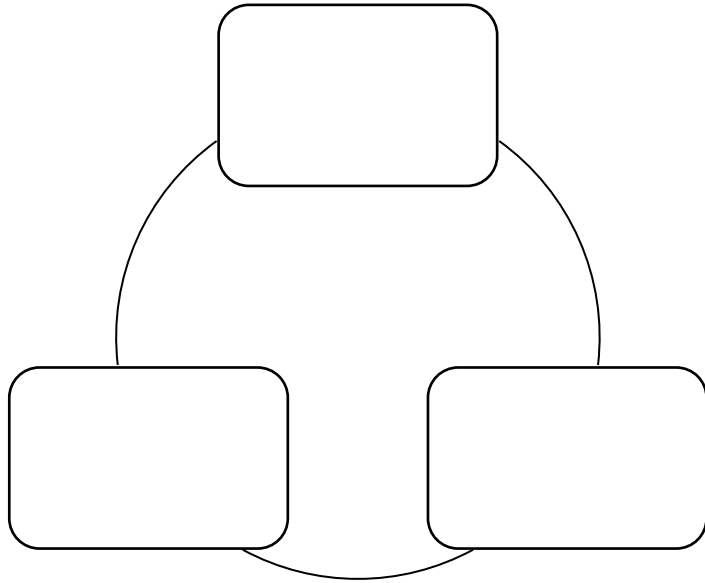
UI prototyping is popular

<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



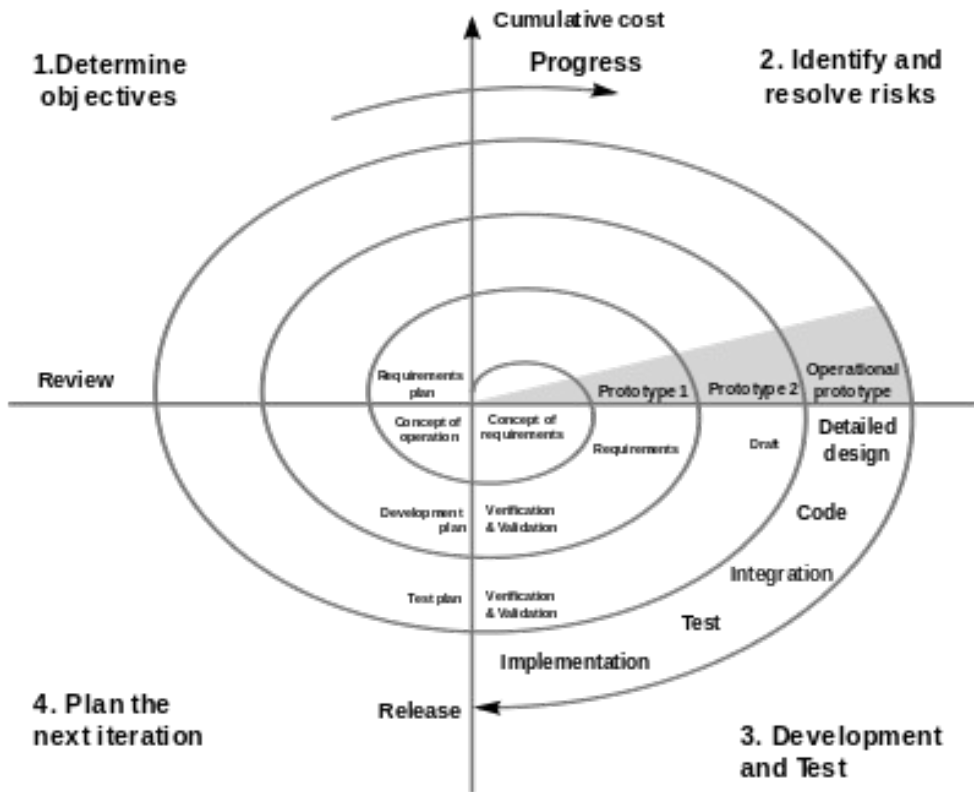
Pros:

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

Cons:

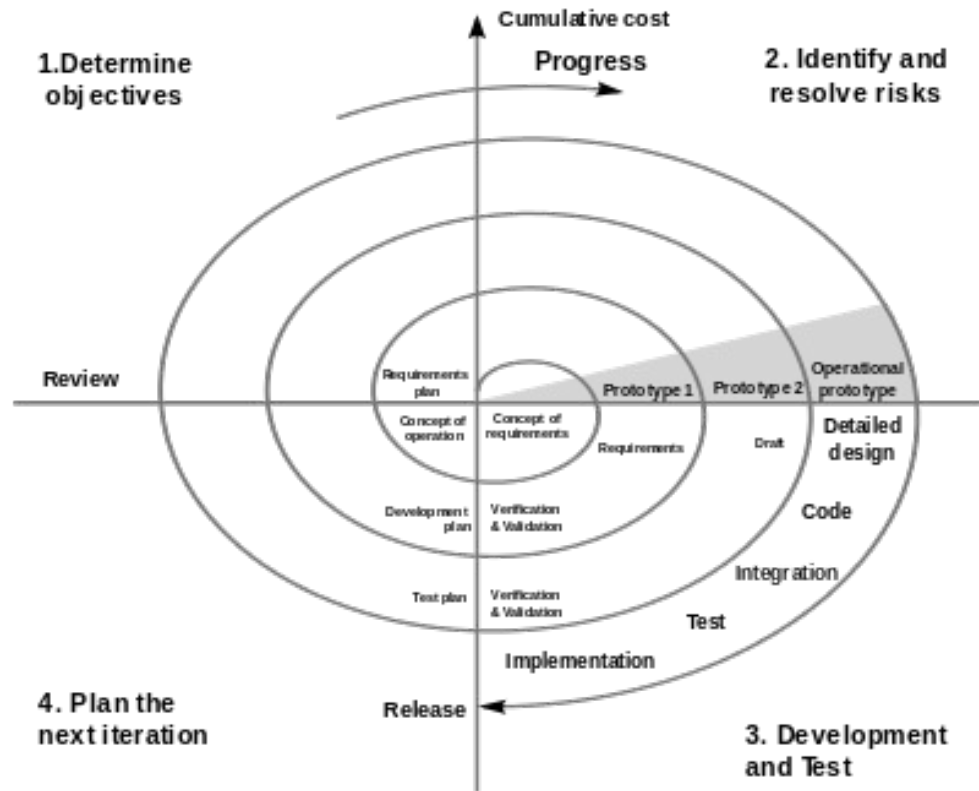
- Time/cost for developing may be high
- Hard to commit what will be delivered and when
- May end up evolving a poor choice (limit thinking holistically)

SDLC: Spiral Model



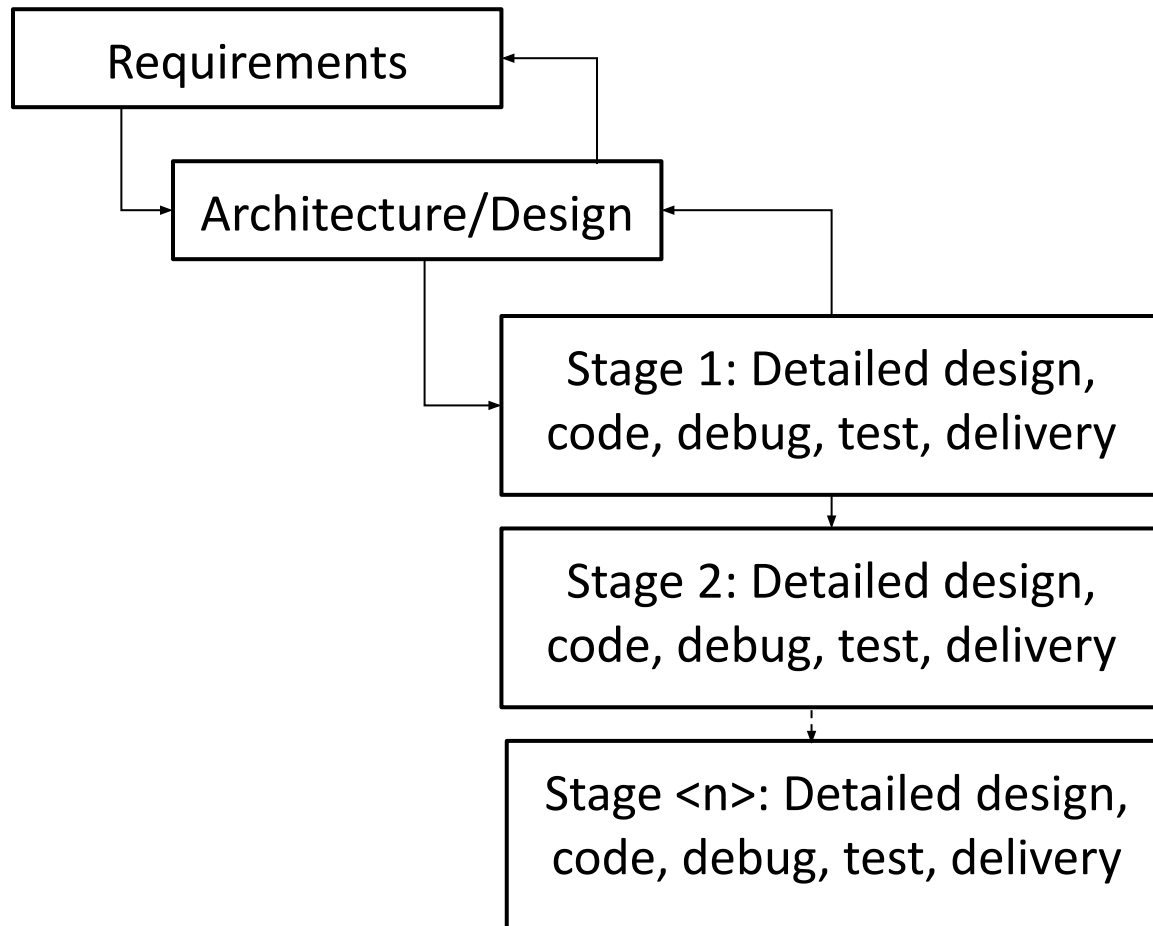
- Incremental/iterative model
- Iterations called spirals
- Repeat these activities:
 - Determine objectives (reqs)
 - Risk analysis
 - Develop and test
 - Plan
- Phased reduction of risks (address high risks early)

SDLC: Spiral Model



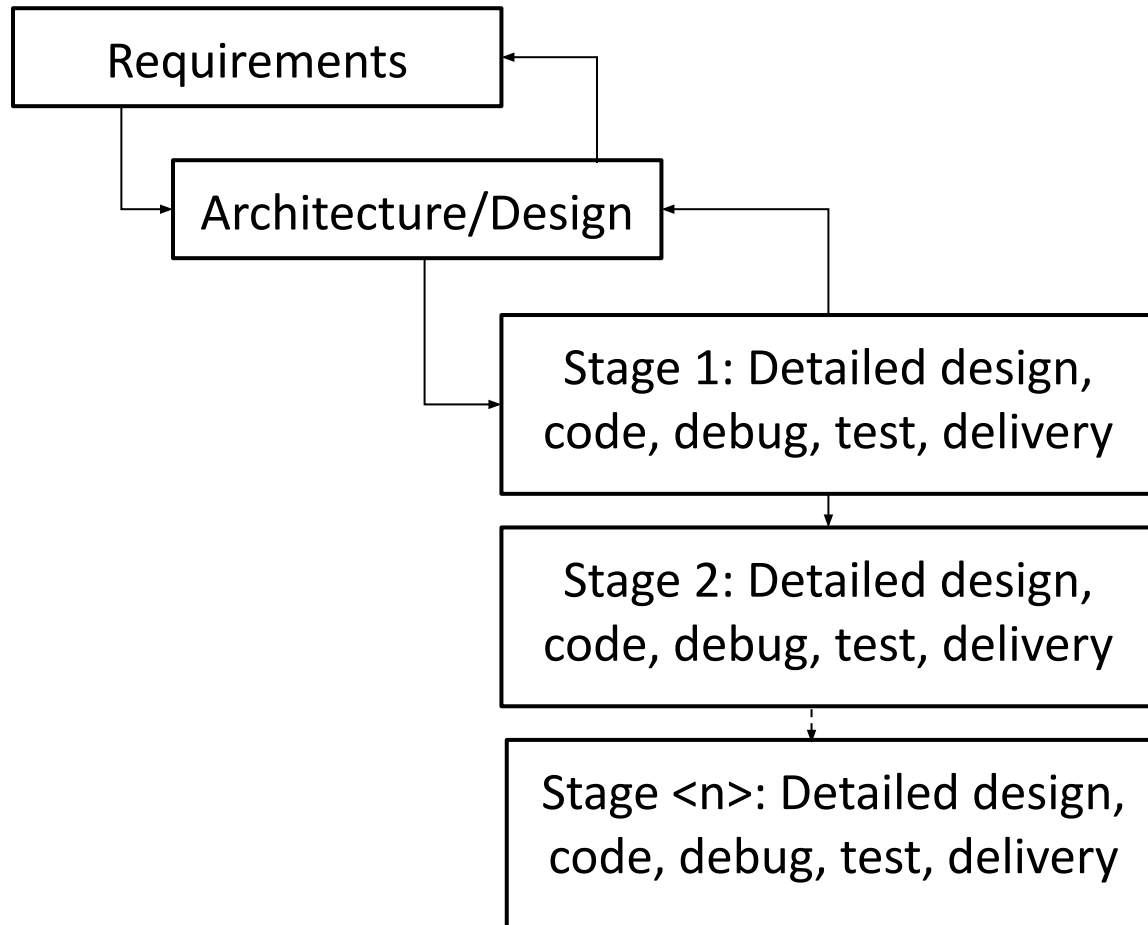
- Interesting to us as it's a **precursor to agile models**
- Software development is based on **iteration**, using “**risk reduction**” as the criteria to prioritize activities at each iteration

SDLC: Lots of variants 🧠 - Staged Delivery



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

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

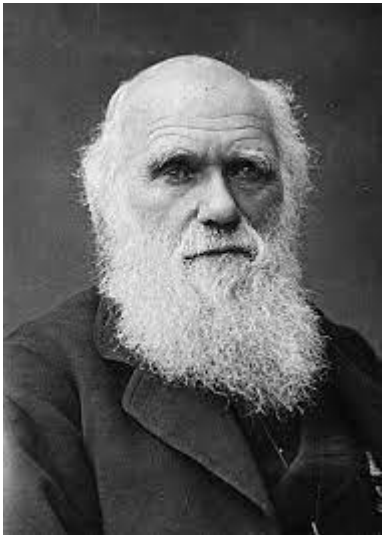
Today's Outline

- Quick introduction
 - 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)
- 

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

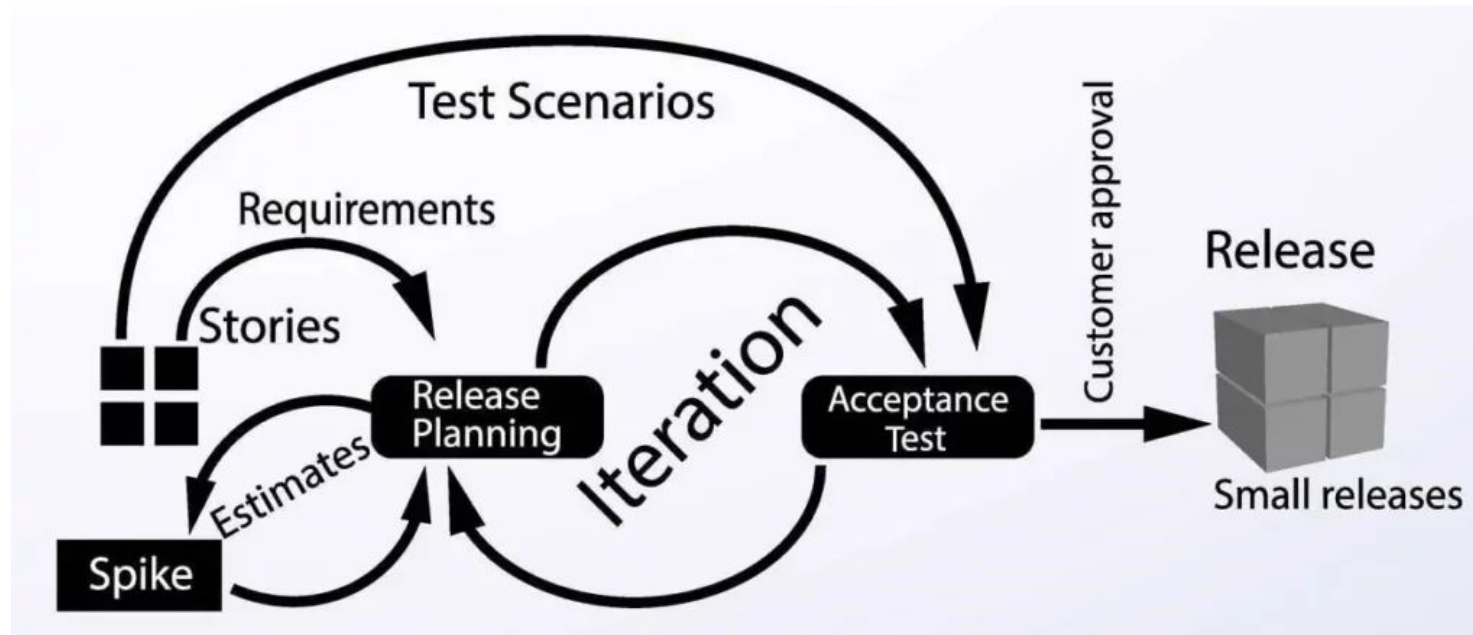
“Agile software development” is a general term for frameworks and practices outlined in the Agile Manifesto

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

<http://agilemanifesto.org/principles.html>

Agile SDLC: Extreme Programming (XP)



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

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

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?

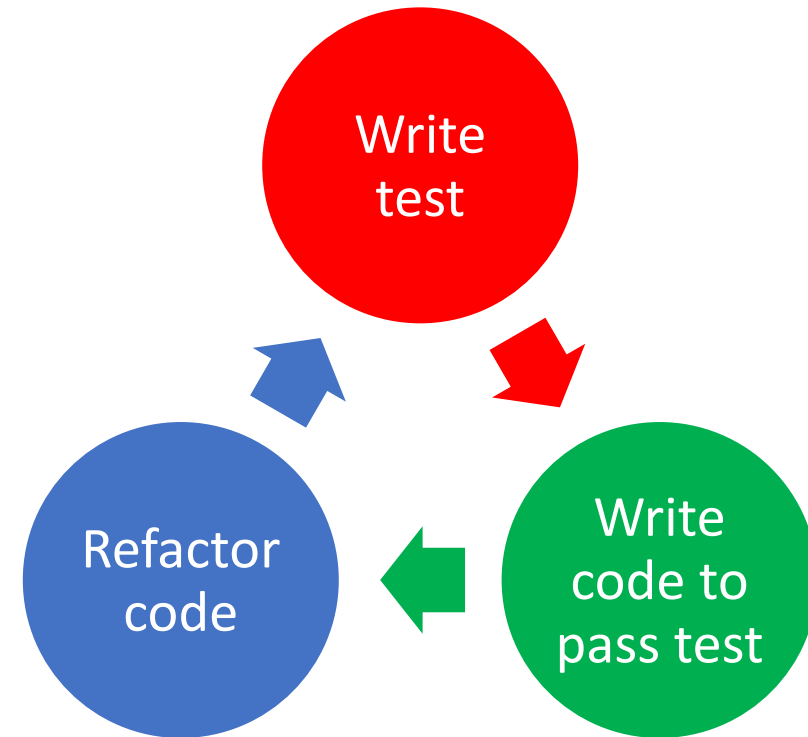


XP Practice: Test driven development

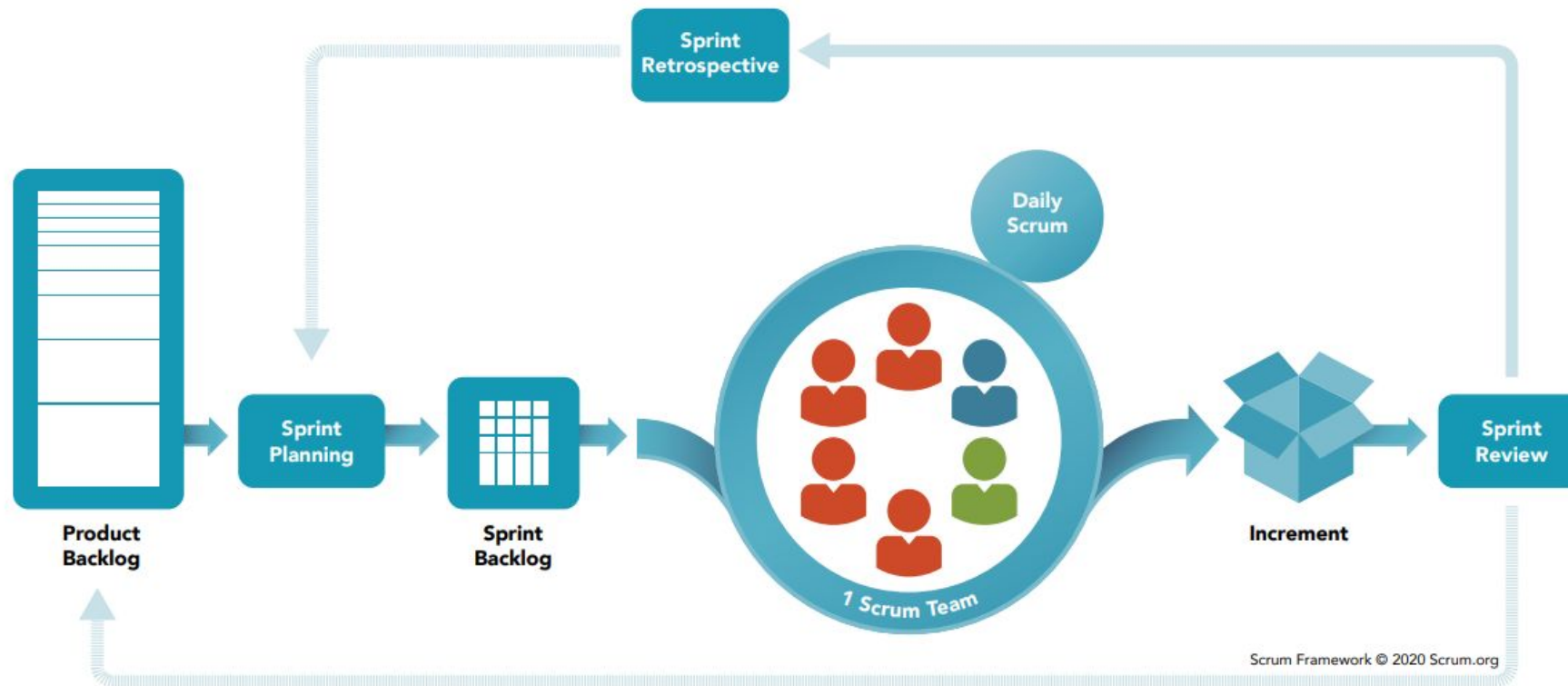
Write tests based on the requirements - before the production code is even written - and then develop code to make the tests pass

Tests run early and often

Thoughts?

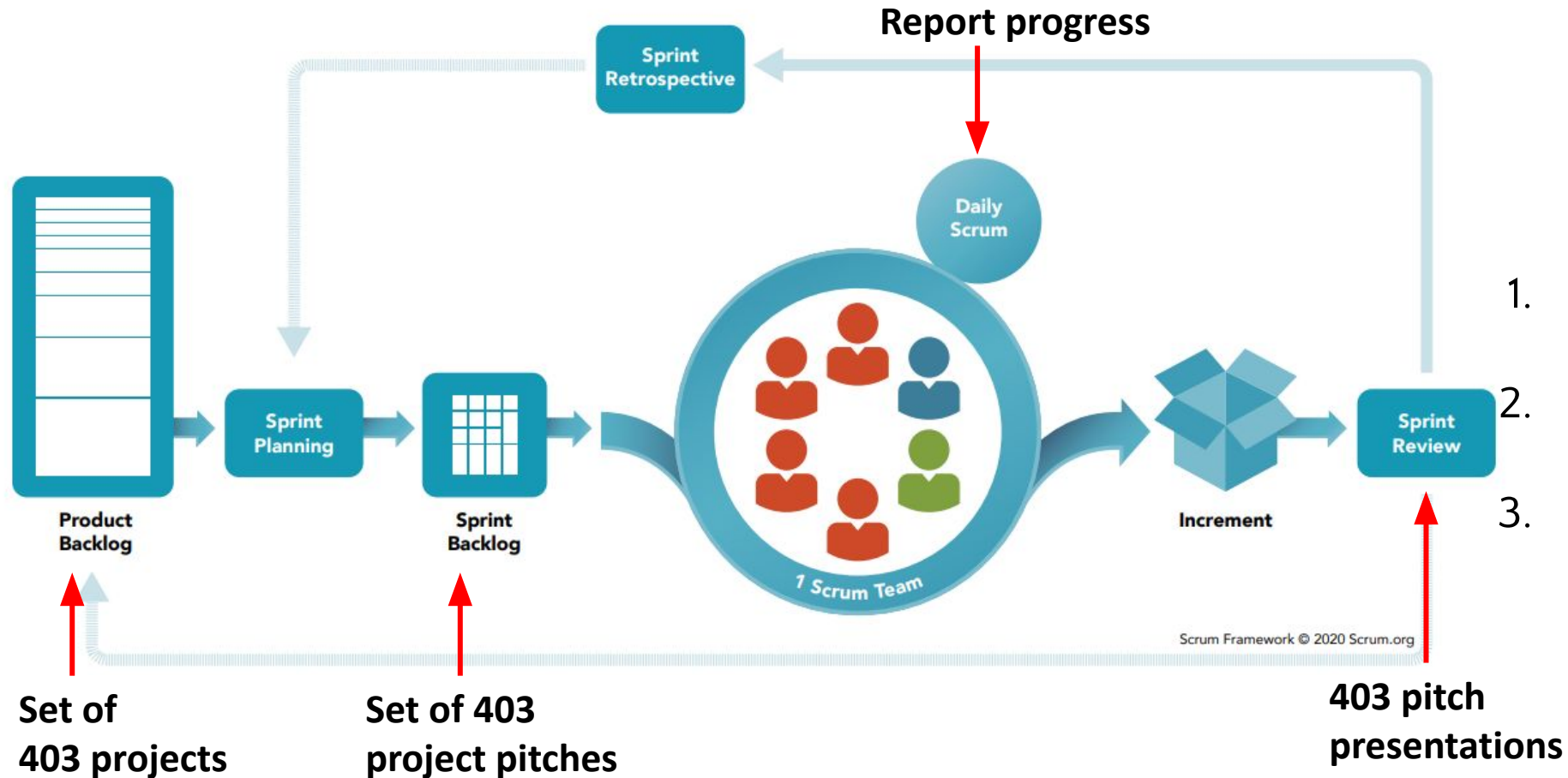


Agile SDLC: Scrum



- Many analogies with XP
- Scrum focuses on management and productivity
- XP addresses software quality and engineering techniques

Shall we try a daily standup?



Daily Standup

Answer 3 questions

1. What did I accomplish yesterday
2. What am I planning for today
3. Am I blocked on anything

Agile Summary

Pros

- Flexibility (changes are expected)
- Focus on quality (continuous testing)
- Focus on communication – with customers – with team

Cons

- Requires experienced management and skilled developers
(e.g., responsible, proactive, communicate well)
- Prioritizing requirements can be difficult when there are multiple stakeholders
- Needs customer to be flexible in delivery (what / when)

What SDLC would you pick and why?

<http://tinyurl.com/cse403-sdlc>



- A control system for anti-lock braking in a car
- A hospital accounting system that replaces an existing one
- An interactive system that allows airline passengers to quickly find replacement flights
- New innovative but tbd features for a social media app
- Your 403 class project

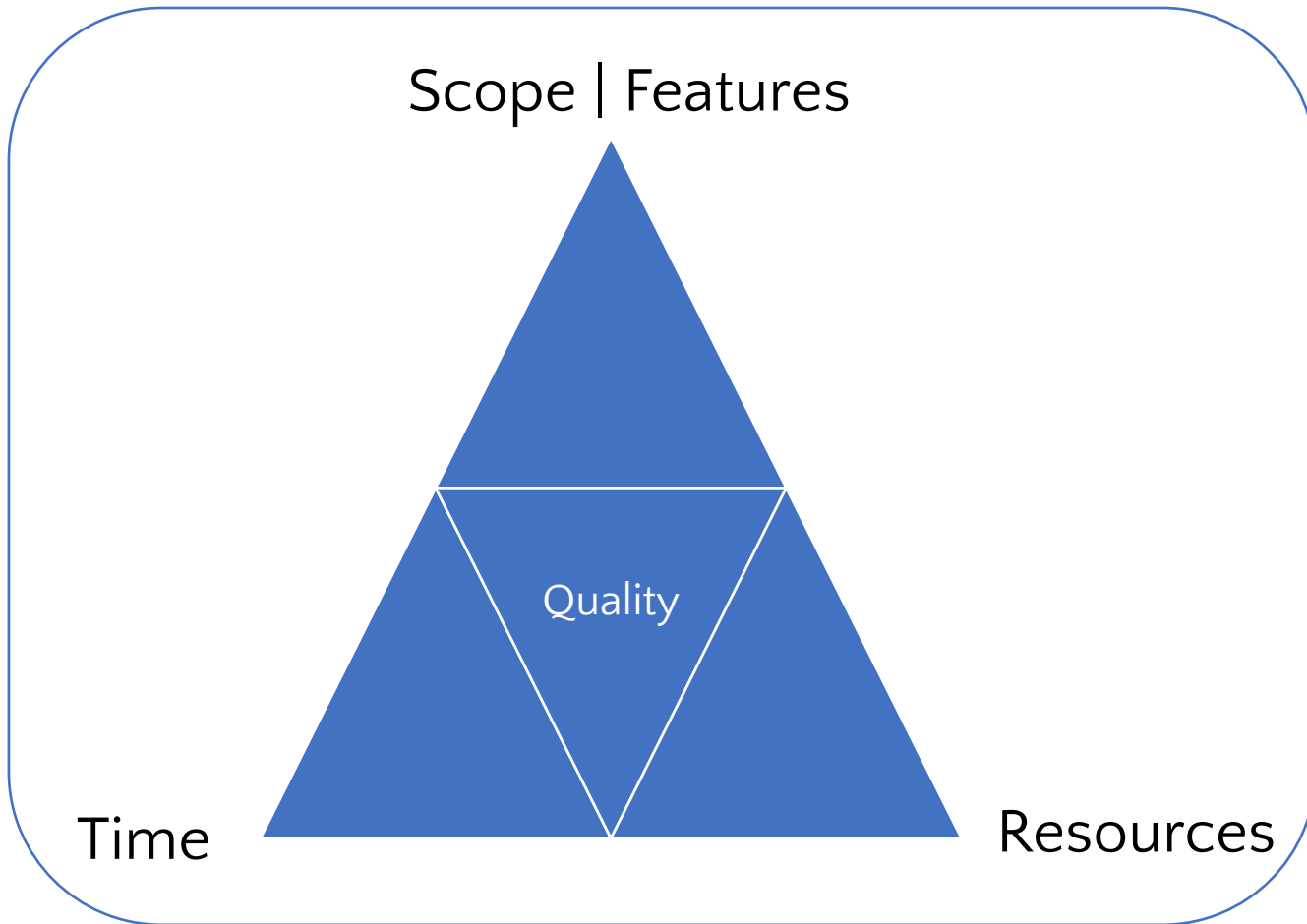
Why are there so many SDLC models?!

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

Questions?

Triangle – project management tool

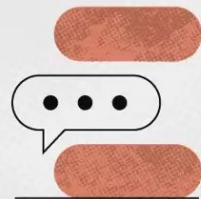


- Software projects must balance what's delivered, when, and with what resources
- When there are changes to one axis, at least one other has to adapt
- These are also good considerations when choosing a SDLC model or adapting to a changing environment

Elevator Pitch

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.

A foolproof elevator pitch template



01
Introduce
yourself



02
Present
the problem



03
Present
your solution



04
Share your value
proposition



05
Add a call
to action

You have 2-3 minutes for your project pitch to the class - this is a good example of how it could flow

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

Press Release

Write a mock product press release describing your product

Includes

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