SCRUM

Module 1: Introduction

Overview

- High level introduction to Scrum
  - It’s relationship to other frameworks and methodologies
  - What Scrum is and what it isn’t
  - How and why Scrum is used
  - The issues Scrum is intended to address, and the ones it is not intended to address
- This is the part of the course that will cover these topics
  - Sets the context for our deep dive into Scrum in the following modules

Origins of Scrum

- Scrum was not originally intended for software development
- Origin was concepts from manufacturing int the 1950s-1980s
  - Very influential: Lean manufacturing & Toyota production system
- In the 1980s, there was a major crisis in software development
  - It was being developed in a big-bang approach
  - Siloed teams (design, development, testing) with one-time hand offs of artifacts
  - NATO software engineering conference in 1968 was held to address the high failure rate of these big-bang projects
  - The conclusion of this and other similar conferences was that in incremental and iterative approach, like Kaizen (continuous improvement) and lean approaches to product development were needed

Origins of Scrum

- In the 1980s and 1990s
  - Companies experimented with using what they called adaptive approaches
- IBM, DuPont, and others experimented with iterative prototyping and empirical process control.
  - Derived from the work mentioned in the previous slide and others (especially lean)
- Characterized by
  - Use of successive prototypes to get feedback on requirements, design and performance
  - Short iterations (one month or less),
  - Cross-functional teams,
  - Daily meetings for synchronization,
  - A prioritized feature list

Origins of Scrum

- In the 1980s and 1990s
  - Companies experimented with using what they called adaptive approaches
- IBM, DuPont, and others experimented with iterative prototyping and empirical process control.
  - Derived from the work mentioned in the previous slide and others (especially lean)
- Characterized by
  - Use of successive prototypes to get feedback on requirements, design and performance
  - Short iterations (one month or less),
  - Cross-functional teams,
  - Daily meetings for synchronization,
  - A prioritized feature list

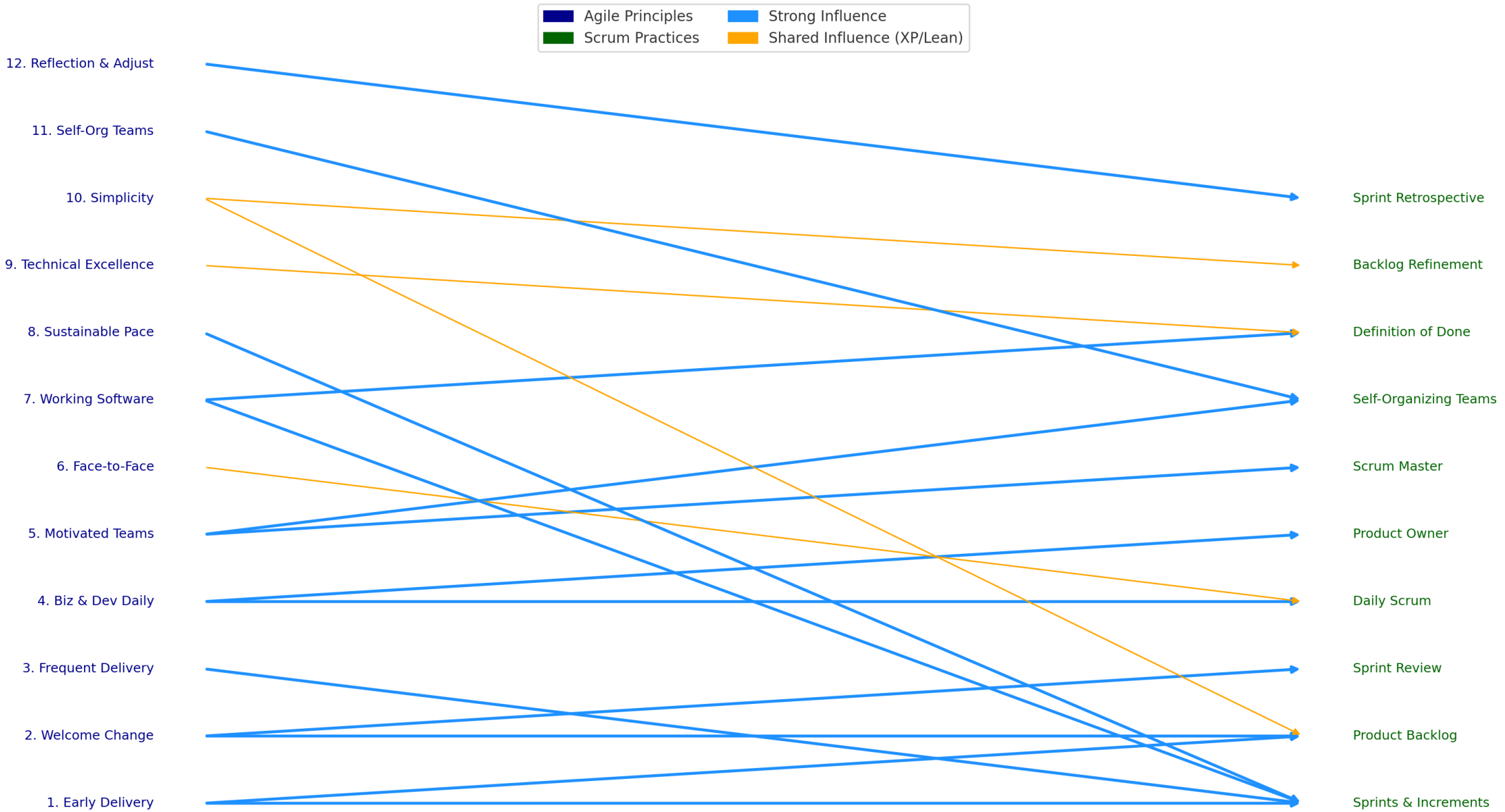
Origins of Scrum

- The term Scrum is first used in 1986 with reference to new product development
  - Hirotaka Takeuchi and Ikujiro Nonaka *“The New New Product Development Game”*
- Jeff Sutherland & Ken Schwaber (1997)
  - Presented the first public paper on Scrum: "SCRUM Development Process."
  - Emphasized:
    - *Empirical process control theory (transparency, inspection, adaptation),*
    - *Lean principles,*
    - *Nonaka & Takeuchi’s knowledge-creation theory (SECI) – the basis for cross functional teams*
- The basic ideas expressed in Scrum started to be adopted
  - Primarily by teams working with highly adaptive types of projects
  - Often smaller teams of developers working closely with the business side

Scrum and Agile

- Agile was the general term adopted in 2001
  - By a consortium of owners of adaptive methodologies that shared a similar approach to development derived from their use of Scrum ideas
    - *eg. Extreme Programming, Feature Driven Development*
  - As a result, many of the features of Scrum were incorporated into the Agile Manifest and the Agile Principles
- Note that Scrum is NOT an Agile methodology, but its concepts were shared among Agile methodology
  - Specifically:
    - *Individuals and interactions*
    - *Working software prototypes*
    - *Customer collaboration and feedback loops*
    - *Responding to change.*

Scrum Influence on Agile Principles



Formalization of Scrum

- 2002 – Scrum Alliance founded to promote Scrum through certifications, communities, and training.
- 2009 – Jeff Sutherland & Ken Schwaber publish "Scrum Guide (1st Edition)"
  - First formal codification of Scrum as a lightweight framework. Defined its roles, events, artifacts, and rules.
- 2010s – Widespread Adoption Beyond Software
  - Scrum principles applied to: Marketing, HR, education, and hardware development.
  - Entire organizations (Enterprise Scrum, Scrum@Scale, LeSS, Nexus).
- 2010 – Scrum.org founded by Ken Schwaber
  - Alternative certifying body with a focus on assessment-driven learning.
- 2017 – Major Scrum Guide Update
  - More emphasis on values: commitment, focus, openness, respect, courage.
- 2020 – Scrum Guide Simplification
  - Roles streamlined (Development Team → Developers),
  - Focus on a single Scrum Team with a shared goal,
  - Removed prescriptive elements to broaden use outside software.

Major Ideas Incorporated into Scrum

- Empirical Process Control (transparency, inspection, adaptation)
  - From Lean manufacturing and scientific method.
- Cross-functional, self-organizing teams
  - From Takeuchi & Nonaka and Lean.
- Iterative & Incremental Development
  - Inspired by early software engineering pioneers.
- Small Batches & Feedback Loops
  - From Lean manufacturing and prototyping.
- Knowledge Creation & Tacit Knowledge Sharing
  - From Nonaka’s work on organizational learning.
- Focus on People and Communication
  - From Fred Brooks and Extreme Programming values.
- Timeboxing
  - Borrowed from project management and software engineering experiments in the 1980s–90s.

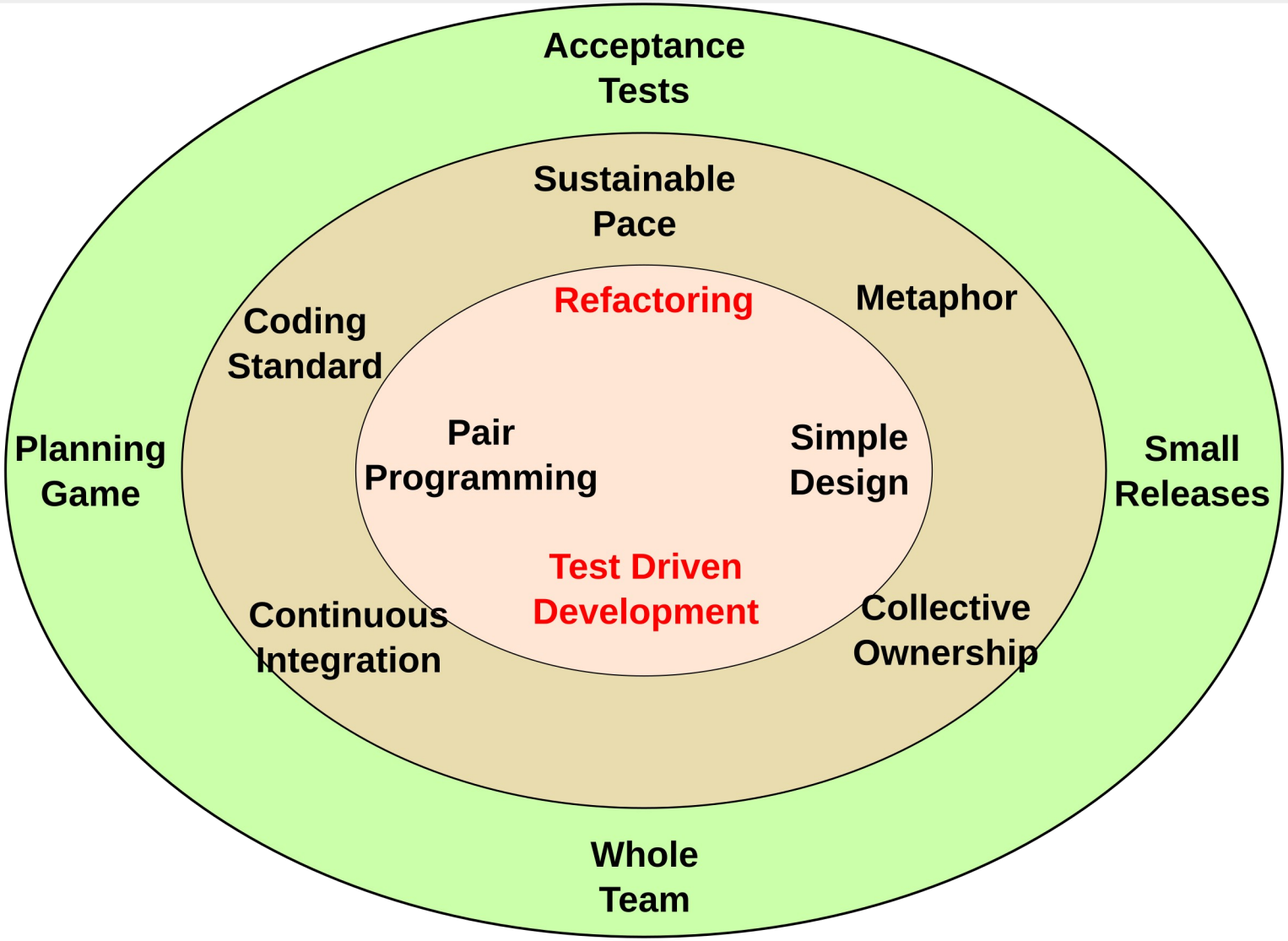
Frameworks and Methodologies

- A common mistake is to consider Scrum to be an Agile methodology
  - It is not a methodology, it is a process framework
  - Failure to have a methodology defined while using Scrum negates the value of using it
- Scrum is process-focused - managing what and when.
- Agile methodologies are practice-focused - managing how the work is done.
- The two complement each other
  - Scrum without a software engineering methodology risks low quality
  - A software engineering methodology without Scrum risks lack of direction and prioritization.

XP Example

- We are not going into any detail about XP
  - We are just using it as an example of an Agile software development methodology for the comparison that follows
- The XP Onion
  - Succinct listing of XP practices
  - Note that they define the actual work to be done
  - Different than other Agile methodologies
- Also notice that we have to separate the “how” from the “what” when we apply Scrum to areas of development other than software.

XP Onion


Acceptance Tests
Sustainable Pace
Refactoring
Metaphor
Coding Standard
Planning Game
Pair Programming
Simple Design
Small Releases
Test Driven Development
Continuous Integration
Collective Ownership
Whole Team

Comparison

- Nature & Purpose
  - Scrum (Framework):
    - *Lightweight container for managing work.*
    - *Defines roles, events, artifacts, and rules.*
    - *Does not prescribe engineering practices — it's about how teams organize and inspect/adapt work.*
    - *"The structure of how the team collaborates."*
  - XP (Methodology):
    - *A set of specific practices for building high-quality software.*
    - *Includes pair programming, test-driven development (TDD), continuous integration, refactoring, and simple design.*
    - *"The actual work done to create the software within the process structure."*

Comparison

- Scope
  - Scrum:
    - *Deals with how teams plan, prioritize, and deliver increments of value.*
    - *Insures transparency, inspection, and adaptation cycles.*
    - *Doesn’t tell you how to code or test.*
  - XP:
    - *Deals with how developers actually build and maintain the software.*
    - *Ensures technical excellence, quality, and adaptability of the product itself.*
    - *Doesn’t define team roles beyond developers.*

Comparison

- Roles & Responsibilities
  - Scrum:
    - *Product Owner: maximizes product value.*
    - *Scrum Master: ensures Scrum is understood & enacted.*
    - *Developers: build increments.*
  - XP:
    - *Customer: provides stories and acceptance tests.*
    - *Developers: write code with XP practices.*
    - *Roles are less structured than Scrum.*
- This is an example of where it is an overlap between the methodology and Scrum
  - This illustrates that we can have customized Scrum roles
  - Scrum is a template that can be tailored, not a set of rules that MUST be followed as is

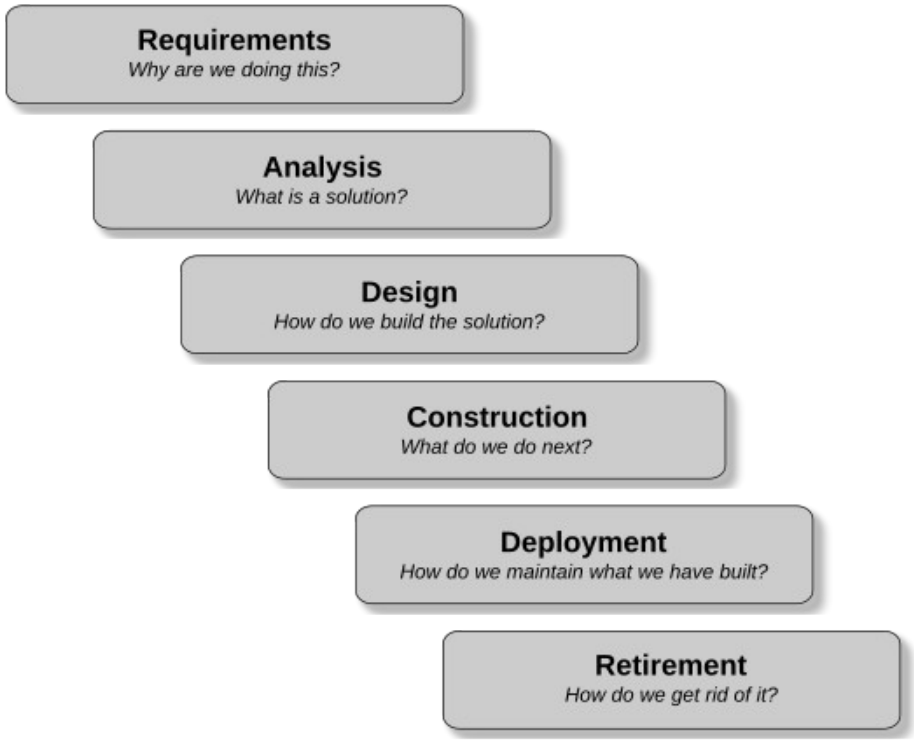
Comparison

- Roles & Responsibilities
  - Scrum:
    - *Product Owner: maximizes product value.*
    - *Scrum Master: ensures Scrum is understood & enacted.*
    - *Developers: build increments.*
  - XP:
    - *Customer: provides stories and acceptance tests.*
    - *Developers: write code with XP practices.*
    - *Roles are less structured than Scrum.*
- This is an example of where it is an overlap between the methodology and Scrum
  - This illustrates that we can have customized Scrum roles
  - Scrum is a template that can be tailored, not a set of rules that MUST be followed as is

The Engineering Process

The engineering process is a description of the logical sequence of natural questions or problems as they occur in any kind of construction process.

The process is universal and can be observed any time we are successfully building or delivering something in an engineering or business context.



Requirements

Basic question that has to be answered during requirements is “*Why are we building this?*”

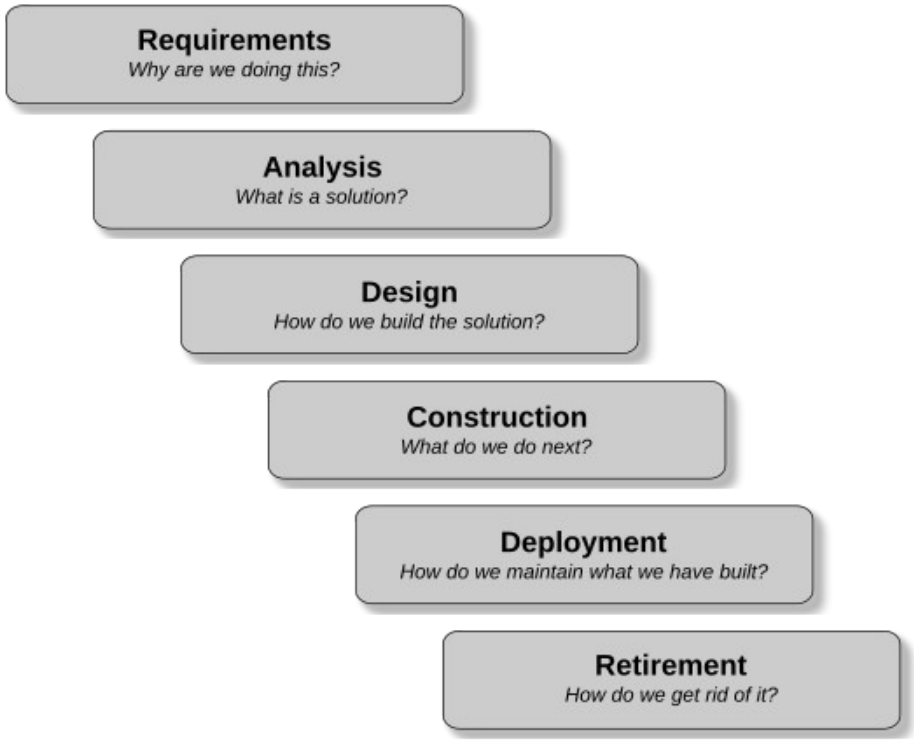
- This is often called the “value proposition”
- Identification of the problem to be rectified

This is generally followed by two other kinds of questions.

“What do the users need what we are building to do for them?”

- The functional requirements (what the stakeholders expect what we are building to do for them)
- The non-functional requirements (what the performance criteria are that what we are building has to meet).

“What are the constraints – budget, time, architecture, etc. – that have to be considered when building the solution?”

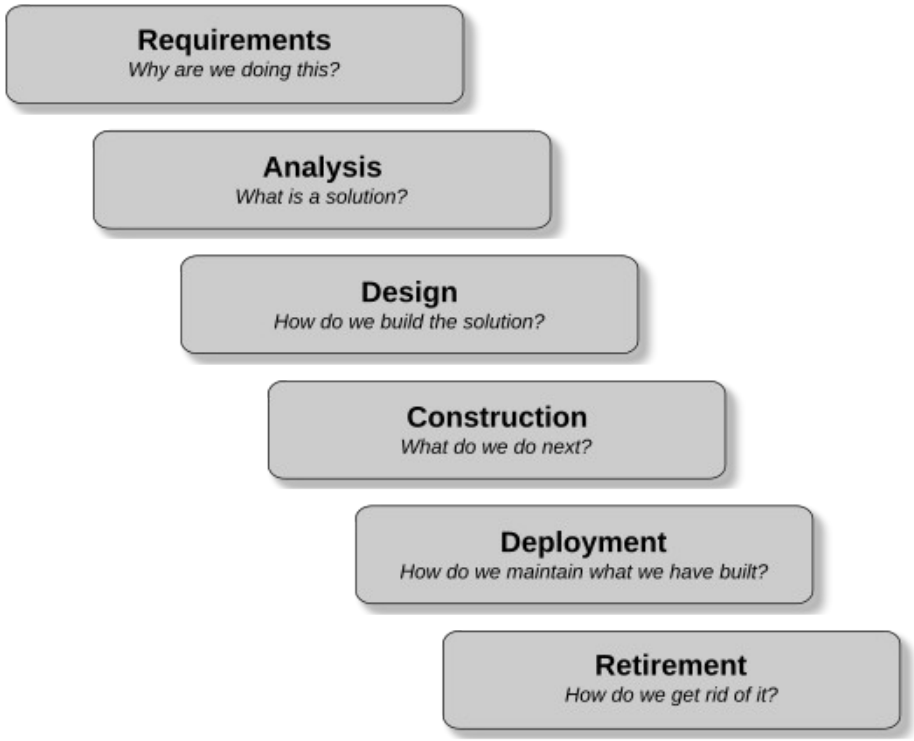


Analysis

In the analysis phase, we find a solution to the problem.

We often formulate the solution in a specification that describes what is to be built.

- The specification serves as a design target
- Whatever design we propose has to produce a product that meets the description of the solution in the specification.
- We may come up with multiple ways to solve the problem which differ in terms of cost, quality and other factors
- But eventually, we need to settle on a single solution.



Design

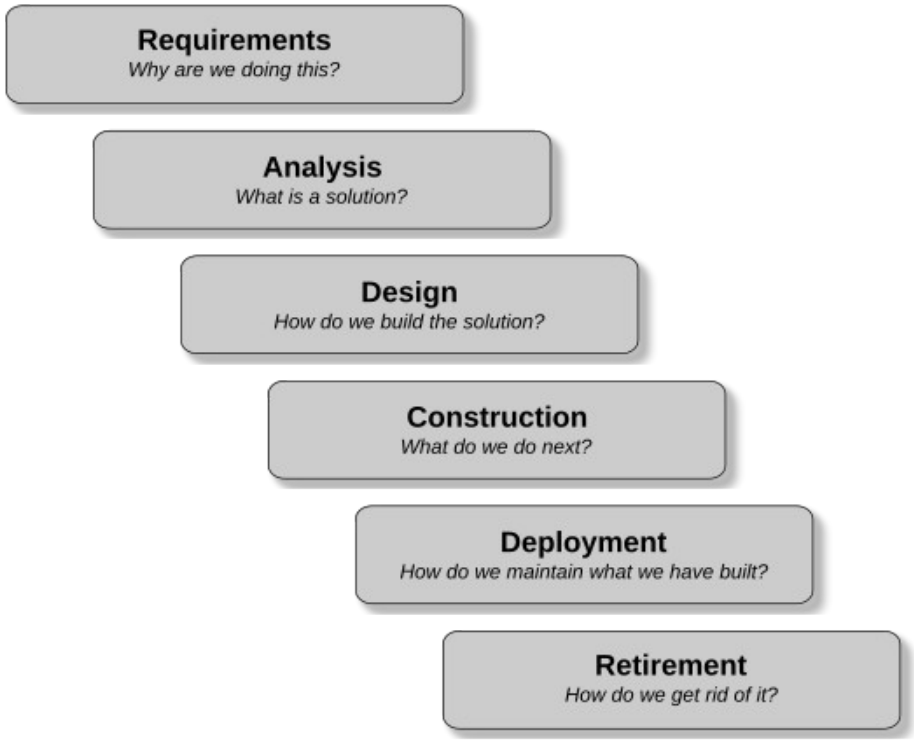
A specification describes a solution but does not tell us how that solution should be built.

A design describes how the solution is to be built given a specific set of resources, capabilities and constraints.

For a single specification, we may have a number of different designs depending on what we have available to work with.

Early prototypes during Analysis

- Will have a design
- Lightweight – more of a plan for experimentation



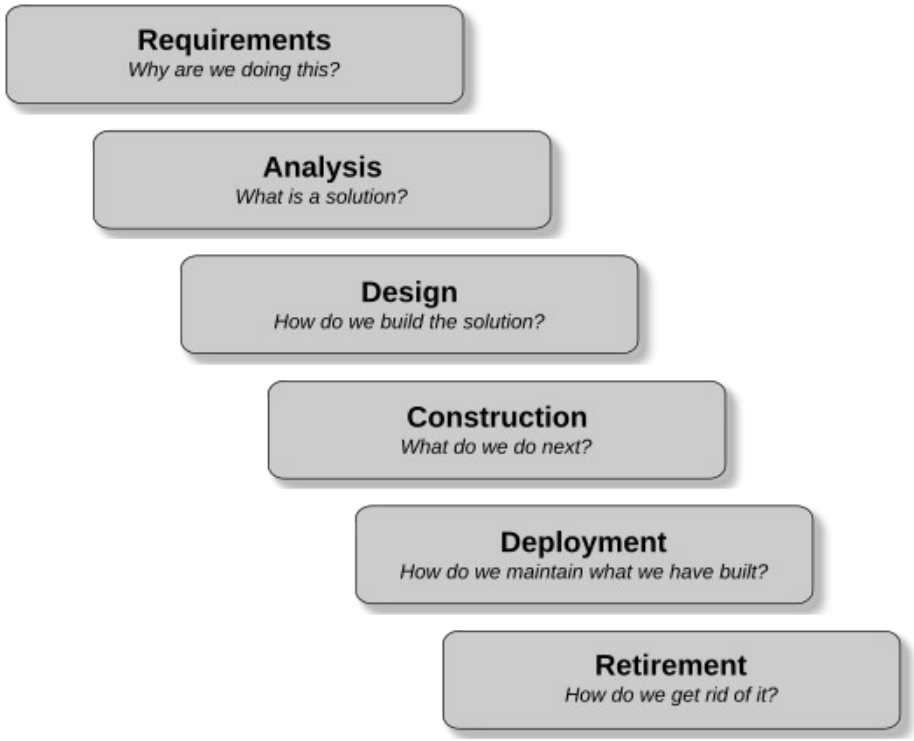
Construction

The product described in the specification is built according to a given design and construction or project plan.

- Having a design alone is not enough for something to be built,
- Especially as the size and complexity of what you are building increases.
- We may also need to build the development infrastructure

Projects also have to use the appropriate construction techniques and plan out the development activities.

- This is the level the Agile methodology is concerned with



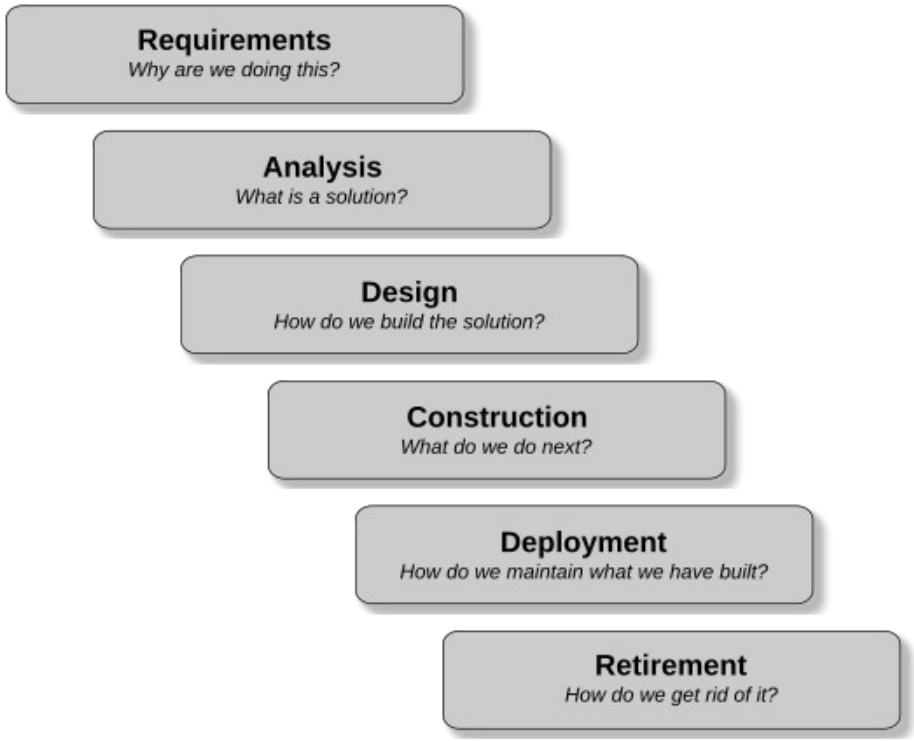
Deployment and Retirement

The scope of Scrum does not cover these two phases

Scrum deals with the construction of software

The processes that support monitoring, updates and retirement are handled by other process frameworks

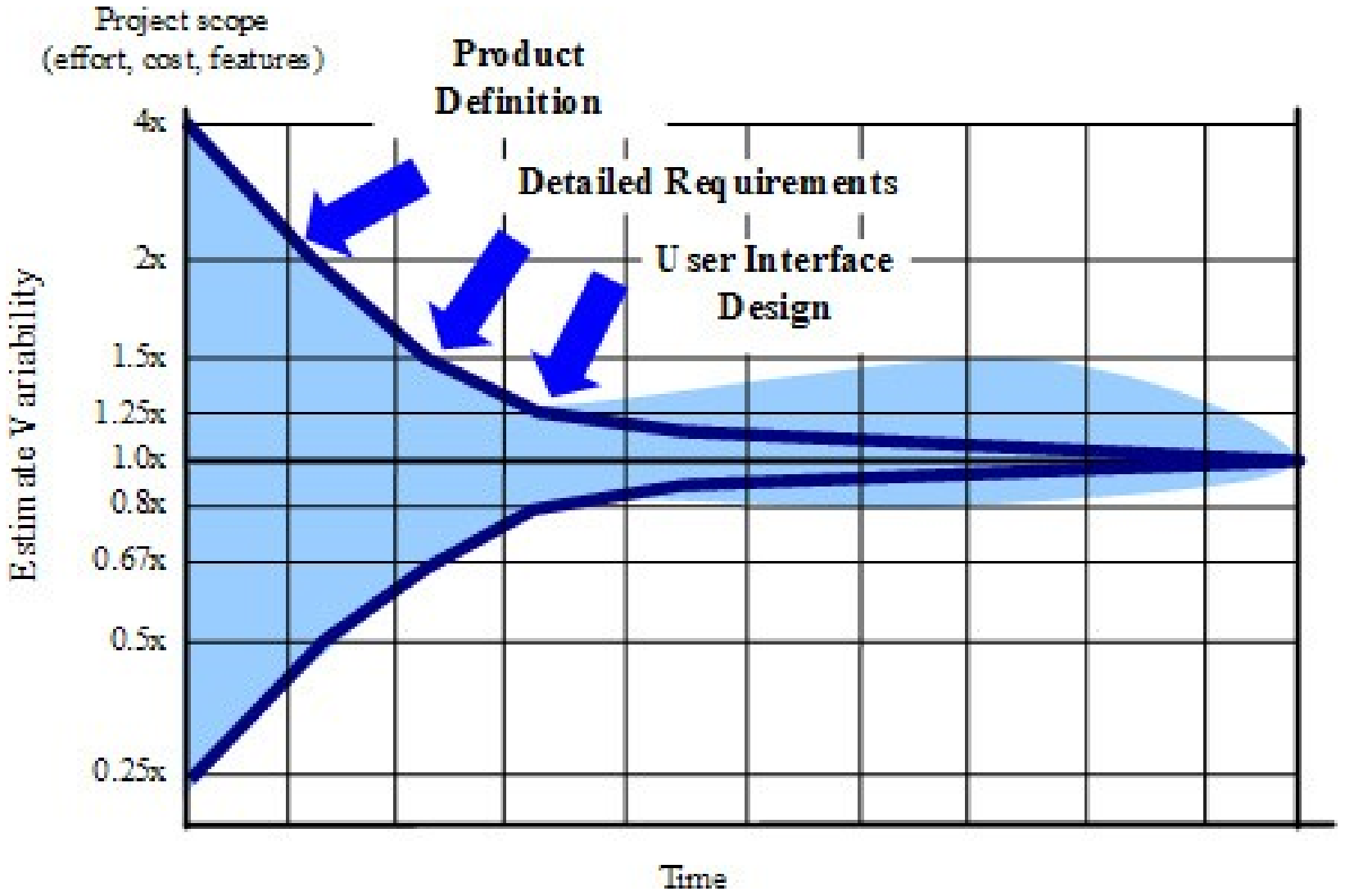
- Eg DevOps, CICD, release management, application lifecycle management
- However, Scrum can be integrated with those process to help support deployment and retirements



Uncertainty

- A big bang process is not generally possible unless
  - Everything is known about the requirements, possible solutions, designs and potential causes of failure for the application
  - Very common for stable engineering environments like civil engineering
  - Also used for very high risk of mission critical software (eg, nuclear reactor controls)
- However, this is not the majority of product development projects
  - We may not know all the requirements until users see a prototype
  - We may not know if proposed solutions actually solve the problem until we prototype
  - We may not know if designs are feasible until we try them out (reliable, affordable and resilient)
- These unknowns at each stage of the process create “slippage”
  - Unless resolved, typically the project fails.

The Cone of Uncertainty

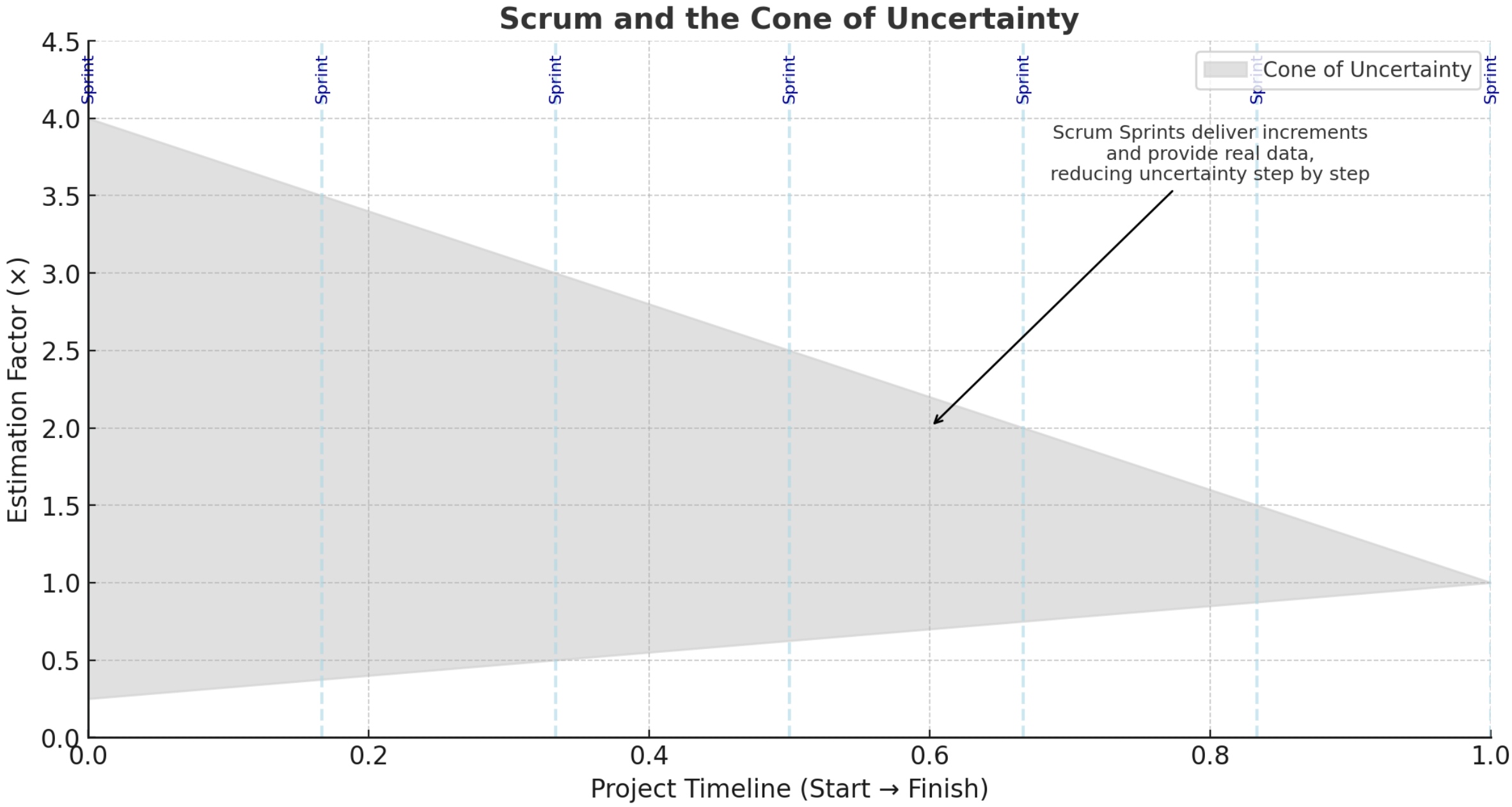


The Cone of Uncertainty

- States that the estimates of time and cost and functionality depend on the amount of information we have about the various stages of the project
  - Initially we start of with wide variance
  - But as we refine requirements, make solution choices, validate designs and try various code implementations, the amount of uncertainty decreases
  - This allows the code to narrow or converge to an accurate estimate
- Unless modified, the estimates will continue to be highly variable
  - At this point the code on uncertainty becomes the cloud of uncertainty
  - Requires incremental changes, evaluation of how well solutions are working
  - And feedback loops to improve estimation

Scrum and Uncertainty

- The Cone of Uncertainty warns us: early estimates are unreliable.
  - Especially if they remain fixed during the development process
- Scrum approaches this problem through:
  - Iterative delivery: reduce uncertainty through working increments.
  - Empirical velocity-based forecasting using evidence, to make estimates more accurate.
  - Backlog reprioritization to optimize value as the cone narrows.
- Scrum actively forces the cone to narrow
  - Studies on IT projects shows that incremental and iterative processes have lower rates of project failure and have more accurate estimates of timelines and budgets


Scrum and the Cone of Uncertainty
Cone of Uncertainty
Sprint
Sprint
Sprint
Sprint
Sprint
Sprint
Sprint
Scrum Sprints deliver increments
and provide real data,
reducing uncertainty step by step
Estimation Factor (×)
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Project Timeline (Start → Finish)

Class Discussions

- At this point in the presentation
  - I want you to start providing me with inputs to improve the class cone of uncertainty
  - The rest of the the course should focus on high value content and activities
  - Over the next several weeks, we will be doing a deep dive into Scrum
  - But we also want to focus on how you are using Scrum, make it relate to your projects
- To do this, I need to know something about:
  - Specific issues you may be having with Scrum
  - Any places we need to review how Scrum is being applied.
  - Where we need to focus our efforts in class
- I will repeat this exercises at the start of the next class
  - In case you think of things over the next week that relate to this.

Questions