Faculty of SET / School of Computer Science

Software Engineering & Project
Lecture 1: Software Process Models

Introductions

James Caddy

- Software Engineering Graduate
- Current PhD Student
- Co-Lecturer and Tutor
- james.caddy@adelaide.edu.au

What is a Software Process?

- A software process is a **structured set of activities** to **produce or maintain** a software product

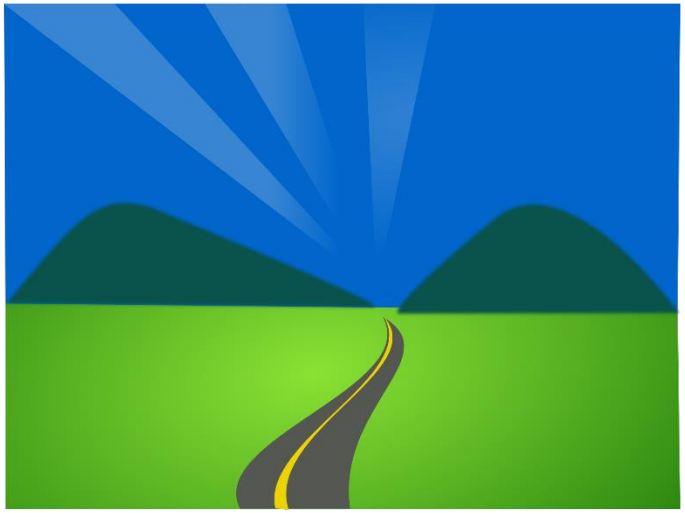
What is a Software Process?

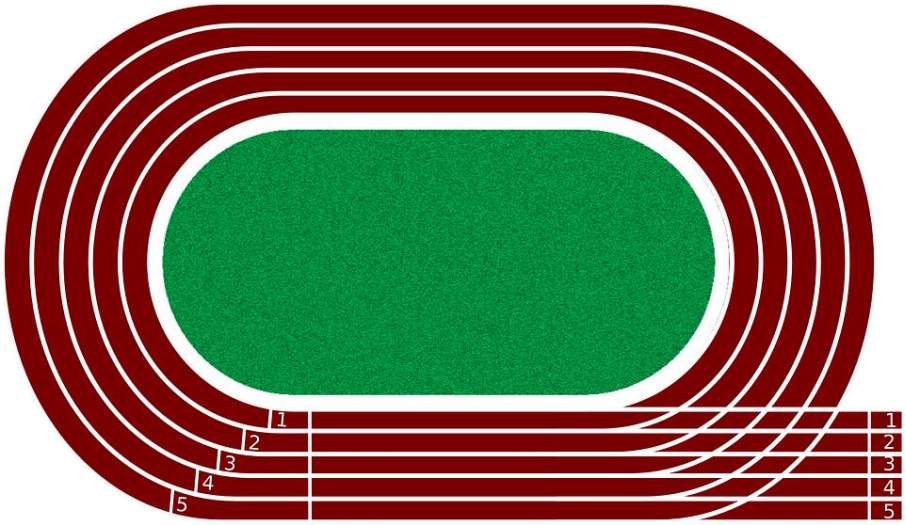
- A software process is a **structured set of activities** to **produce or maintain** a software product

- Examples of fundamental activities in software development:
  - Specification:
    Defining what the system should do
  - Design and Implementation
    Defining the system's architecture and implementing the system
  - Verification and Validation:
    Checking that it does what the customer wants
  - Maintenance:
    Changing the system in response to new customer/market needs

Software Process Models?

- **Abstract** representation of a software development process
- Provide a **description** of the different steps in the process and the corresponding stakeholders
- Useful as a **roadmap** to guide software teams





Generic Software Process Models

- Most software process models favour one of the following **generic models** or paradigms of software development:
  - Waterfall/Big design up front
  - Incremental/agile software development

Generic Software Process Models

- There is **no one-size-fits-all** model
- Each model has **strengths** and **weaknesses**
- In practice, most large systems are developed using a process that **incorporates elements** from different models

More information:
https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc

Is *Water-Scrum-Fall* Reality? On the Use of Agile and Traditional Development Practices

Georgios Theocharis[1], Marco Kuhrmann[1](✉), Jürgen Münch[2], and Philipp Diebold[3]

[1] The Mærsk Mc-Kinney Møller Institute and Center for Energy Informatics, University of Southern Denmark, Odense, Denmark
g.theoharis84@gmail.com, kuhrmann@mmmi.sdu.dk

[2] Department of Computer Science, University of Helsinki, Helsinki, Finland
Juergen.Muench@cs.helsinki.fi

[3] Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany
philipp.diebold@iese.fraunhofer.de

Abstract. For years, agile methods are considered the most promising route toward successful software development, and a considerable number of published studies the (successful) use of agile methods and reports on the benefits companies have from adopting agile methods. Yet, since the world is not black or white, the question for what happened to the traditional models arises. Are traditional models replaced by agile methods? How is the transformation toward Agile managed, and, moreover, where did it start? With this paper we close a gap in literature by studying the general process use over time to investigate how traditional and agile methods are used. Is there coexistence or do agile methods accelerate the traditional processes' extinction? The findings of our literature study comprise two major results: First, studies and reliable numbers on the general process model use are rare, i.e., we lack quantitative data on the actual process use and, thus, we often lack the ability to ground process-related research in practically relevant issues. Second, despite the assumed dominance of agile methods, our results clearly show that companies enact context-specific hybrid solutions in which traditional and agile development approaches are used in combination.

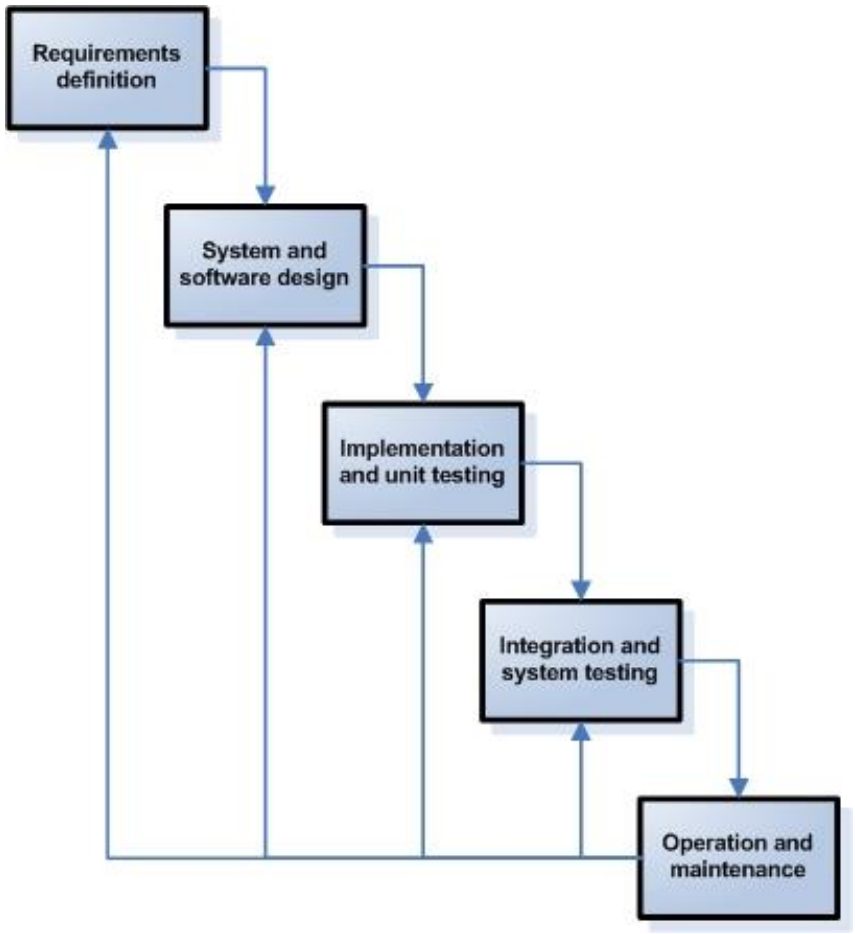
https://link.springer.com/chapter/10.1007/978-3-319-26844-6_11

The Waterfall Model

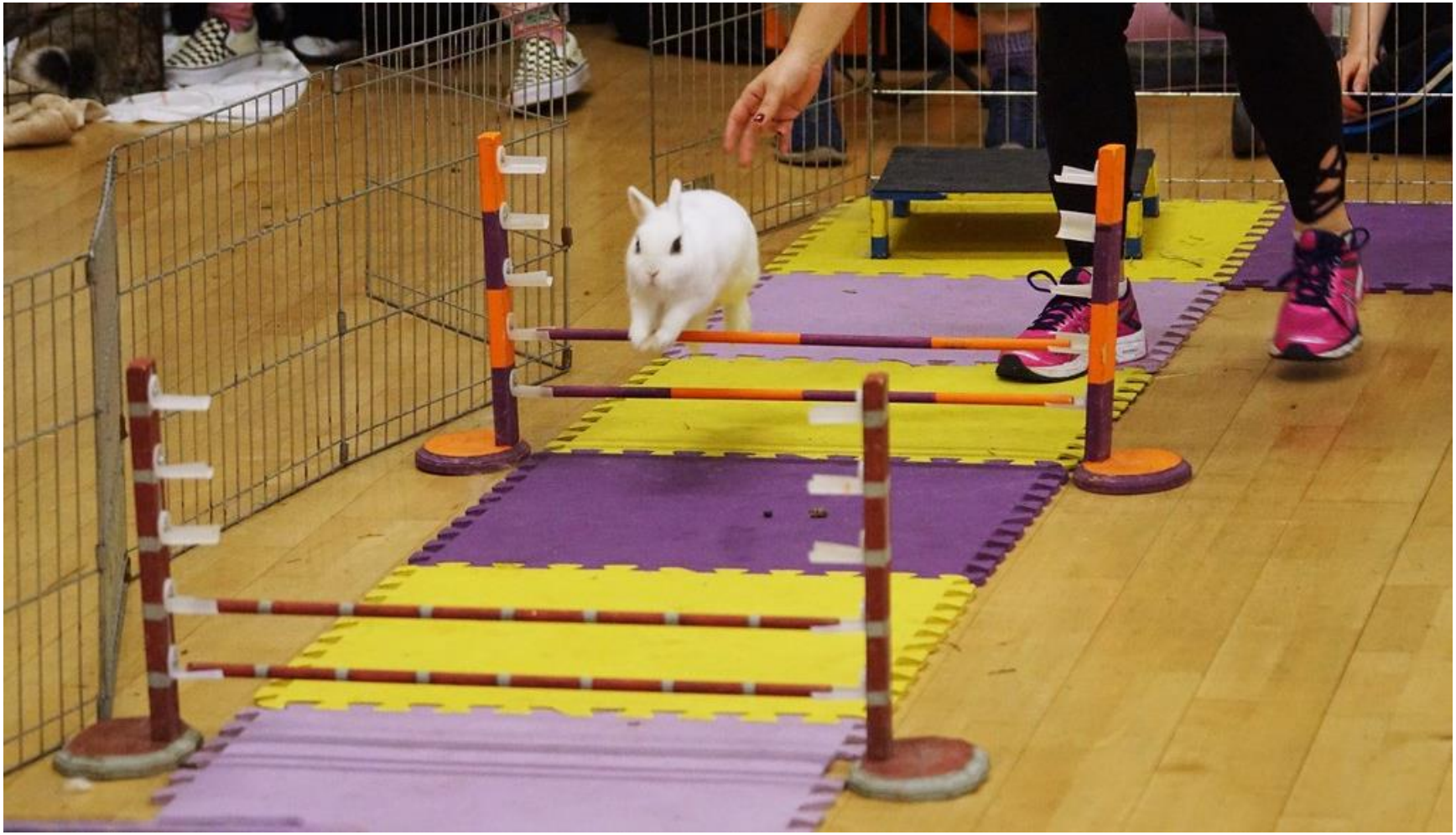
The Waterfall Model

- **Strengths**
  - Aligns to the systems engineering process (hardware)
  - Thorough documentation
- **Weaknesses**
  - Inflexible to changing requirements
  - Late discovery of technical problems due to sequential order

- This model is **suitable when the requirements are well-understood** and changes will be fairly limited during the design process.



Agile Development

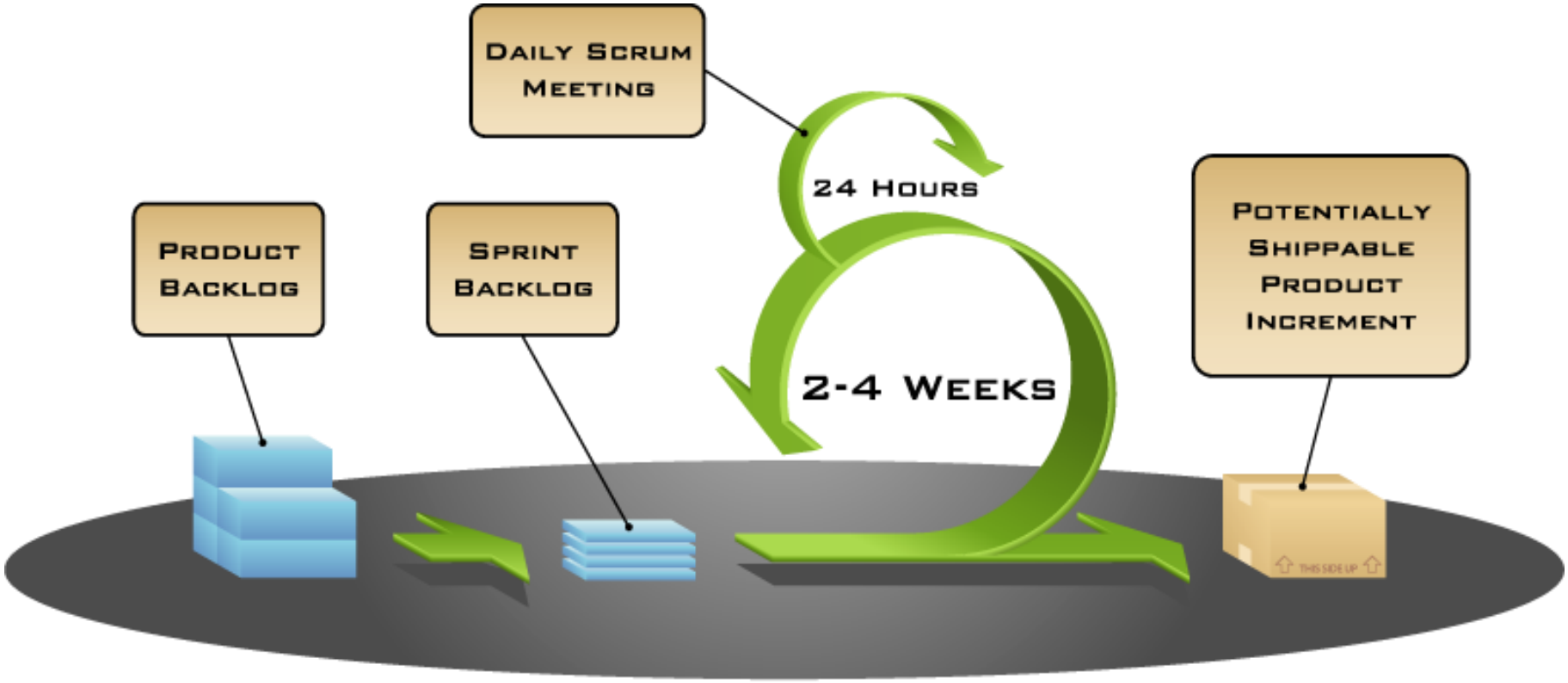


Agile Development

- Goal: Requirements and solutions are supposed to **evolve** through **collaborative effort** of teams and customers
- The model advocates…
  - Adaptive planning
  - Iterative development
  - Early delivery
  - Continuous improvement
- …to be able to **rapidly and flexibly respond to change**
- Term "agile" gained popularity with agile manifesto (2001)
- Popular agile frameworks:
  - Scrum
  - Kanban

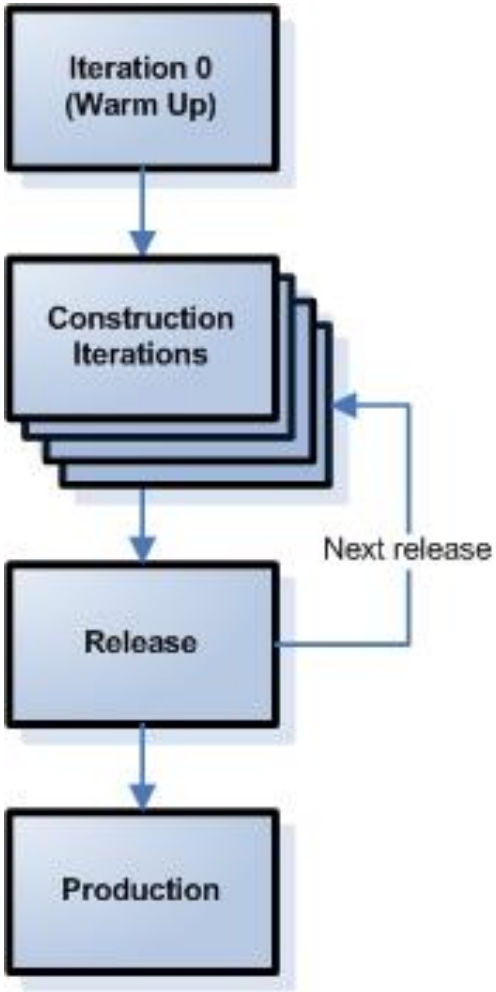
More information:
https://en.wikipedia.org/wiki/Agile_software_development

Agile - Scrum


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment

Agile Development

- **Strengths**
  - Extremely **responsive** to change
  - Working software produced very **early**
- **Weaknesses**
  - Highly **dependent** on customer involvement
  - Requires a **high performing** team



Agile Manifesto

In **2001** in Utah, USA, a group of 17 experienced software developers got together to **discuss** software development **methods** that were **lightweight and easy to implement.**

https://setandbma.wordpress.com/2012/03/23/agile-history/

https://twitter.com/agilemanifesto

Agile Manifesto

- States values and principles:
  - "***Customer collaboration*** *over contract negotiation*"
  - "***Individuals and interactions*** *over processes and tools*"
  - "***Working software*** *over comprehensive documentation*"
  - "***Responding to change*** *over following a plan*"

https://agilemanifesto.org/

But…

Quora

Search for questions, people, and topics

Software Engineering at Google | Software Engineering and Computer Programming | Agile Software Development | Software Developers | Computer Programmers | Software Engineering | Google (company) | Web Development | Computer Programming

Why do some developers at strong companies like Google consider Agile development to be nonsense?

http://qr.ae/QKWNo

92 Answers

David Jeske, Computer Engineer, former Google Engineer Director.
Updated Dec 10, 2016 · Upvoted by Tom Keane, Staff Software Engineer at Google (2008-present) and Fabio Seixas, Coding the Web since 1995 · Author has **61** answers and **575.7k** answer views

As a former Google Engineering Director, I can offer some insight.

Agile means many things to many people. The simple high level Agile Manifesto is something I think is close to the way Google engineers think about software development.

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

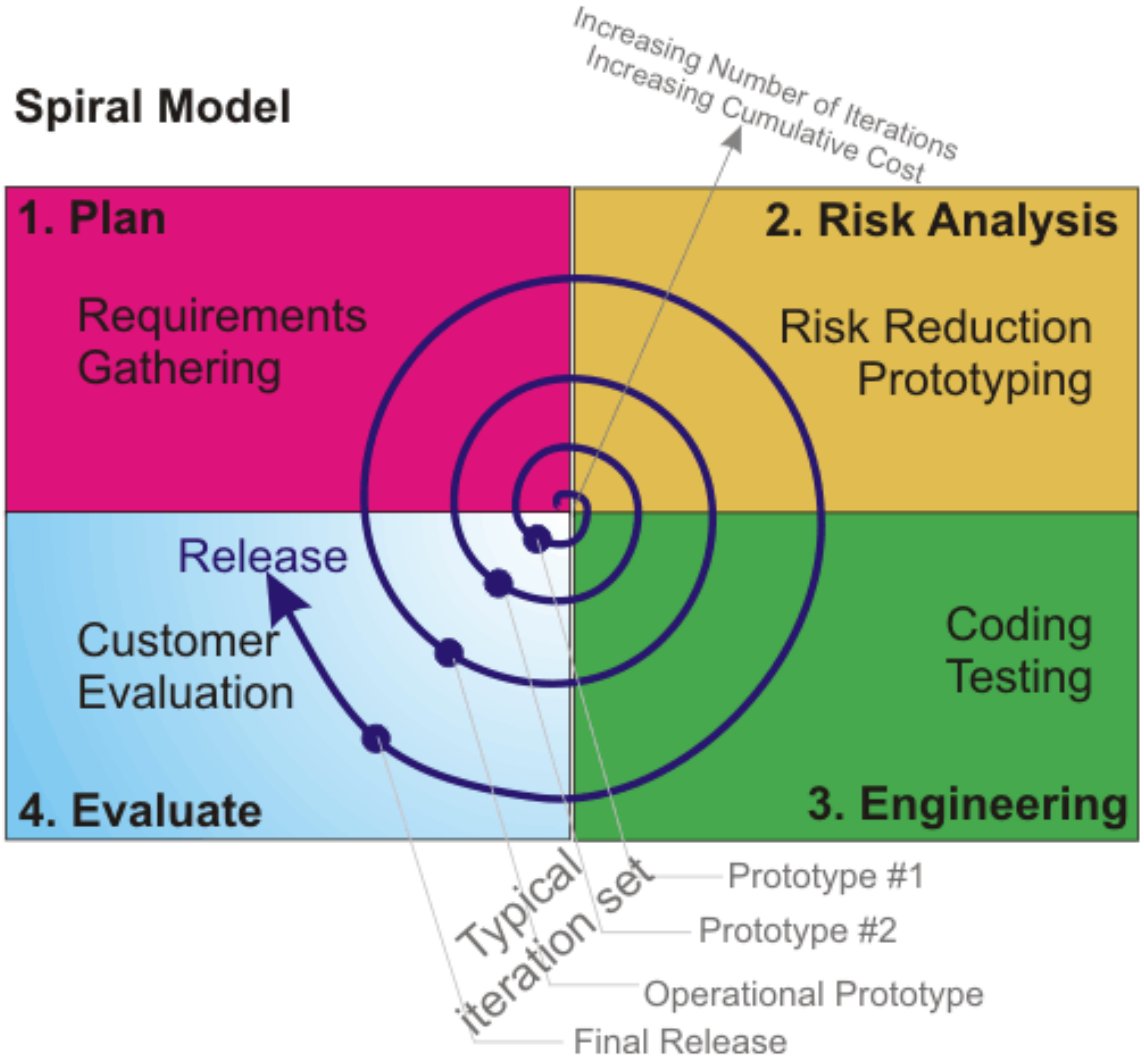
https://www.quora.com/Why-do-some-developers-at-strong-companies-like-Google-consider-Agile-development-to-be-nonsense

The Spiral Model

- Rather a **meta-model**
- Process is represented as a **spiral** rather than as a **sequence** of activities with backtracking
- Each **loop** in the spiral represents a **phase** in the process
- **No fixed phases** such as specification or design, **loops** in the spiral are **chosen depending on what is required**
- **Risks** are explicitly assessed and resolved throughout the process

https://en.wikipedia.org/wiki/Spiral_model

The Spiral Model



Software Process Models in Practice

- In practice, software organisations may **combine aspects of generic process models** to meet the specific needs of their projects
- In many cases the choice of process model is also **constrained** by a range of other factors:
  - Contractual or regulatory requirements
  - Experience and familiarity with the process model
  - Process models used by related projects
  - Team experience and abilities

Scrum

Scrum

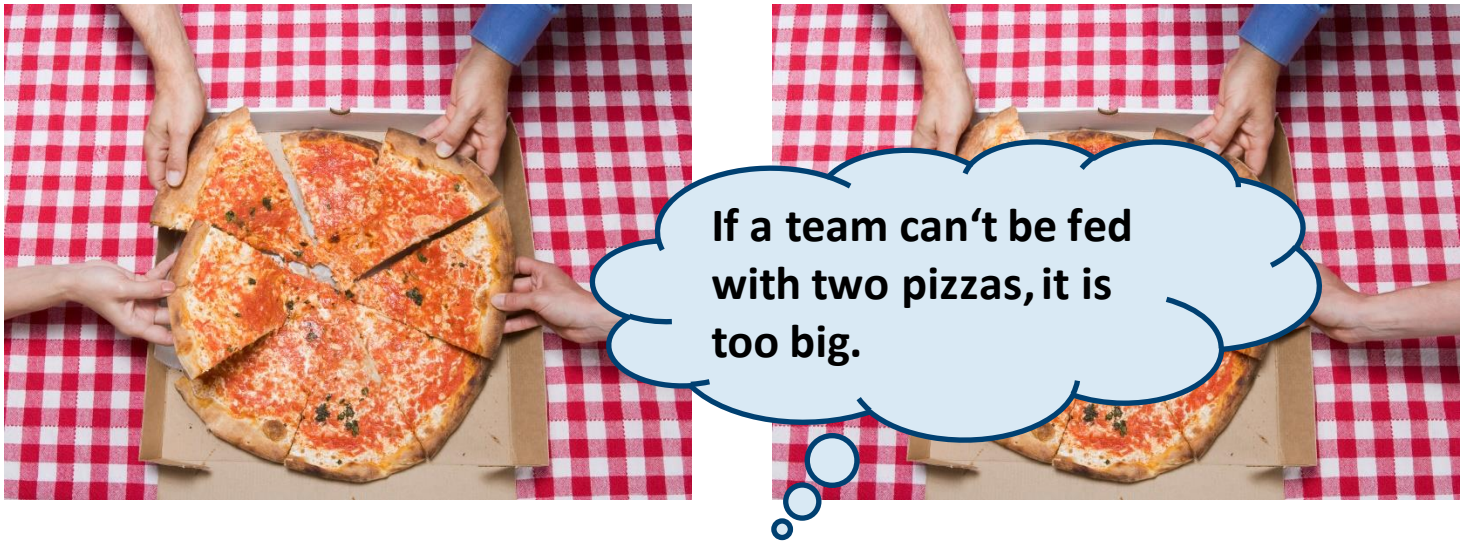
Scrum is an **agile process framework** that focuses on managing iterative development.

Scrum

Rugby

A scrum is a method of restarting play in rugby that involves players packing closely together with their heads down and attempting to gain possession of the ball. Wikipedia

Scrum: Team Organization



- "2 pizza" team size (Jeff Bezos, 4 to 9 people)
- Scrum inspired by frequent short meetings
  - 15 minutes every day at same place and time
  - Book recommendation: *The Elements of Scrum* by Chris Sims and Hillary Louise

https://buffer.com/resources/small-teams-why-startups-often-win-against-google-and-facebook-the-science-behind-why-smaller-teams-get-more-done

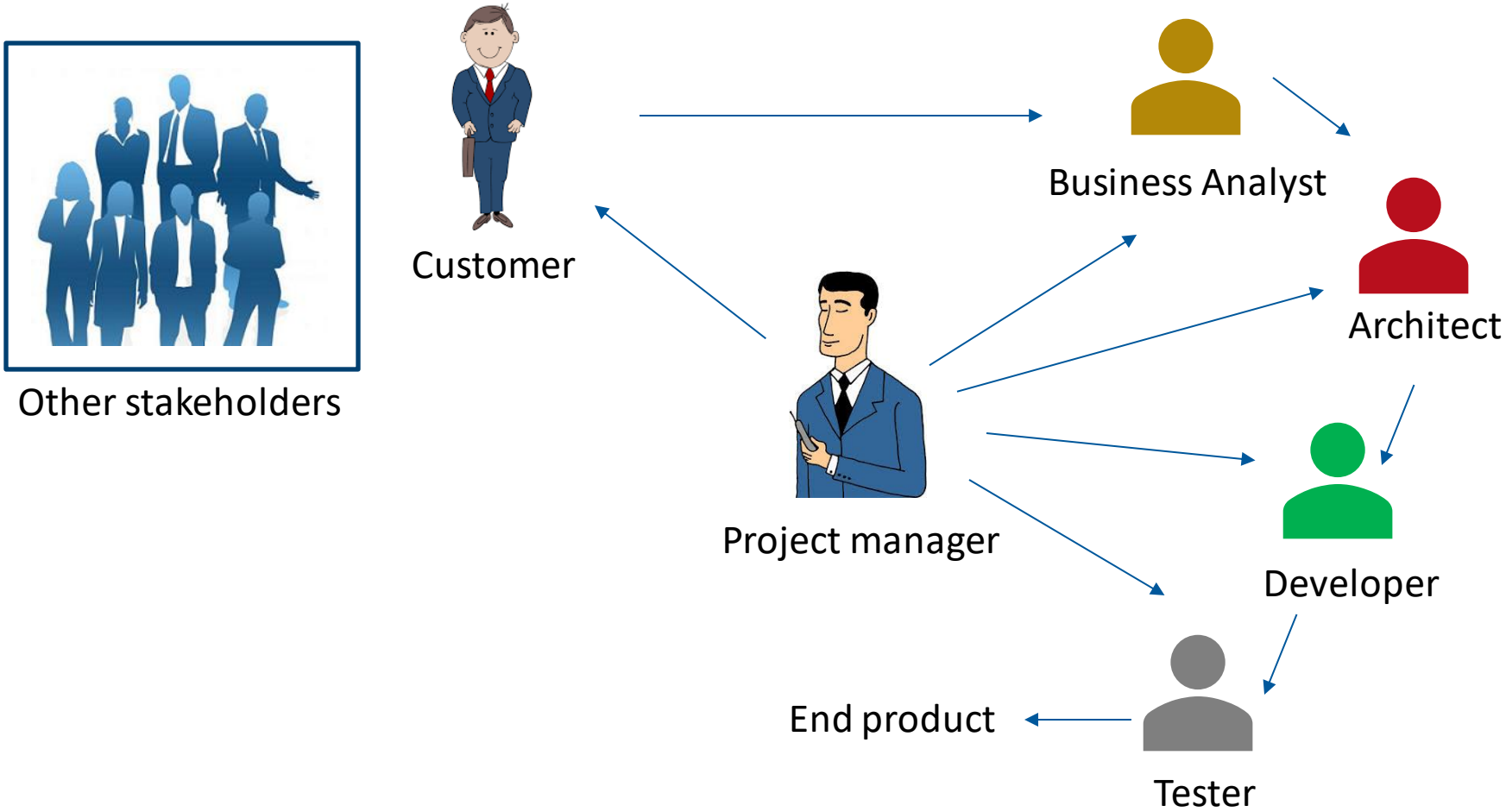
Scrum Roles

- **Team:**
  - 2-pizza size team that delivers software
  - Development organized in "sprints"
  - Sprint:
    - Time after which the product is in a stable, releasable form
    - Software not necessarily released after sprint
    - Unit of iteration
- **Scrum Master:** A team member who
  - Acts as **buffer** between the team and external distractions
  - Keeps team **focused** on task at hand
  - Enforces team **rules** (e.g., coding standards)
  - **Removes impediments** that prevent team from making progress
  - Not a manager or team leader, but a **facilitator**!

Scrum Roles

- **Product Owner:** A team member (not the Scrum Master) who represents the **voice of the customer** and prioritizes **user stories** that capture requirements.

Traditional Way

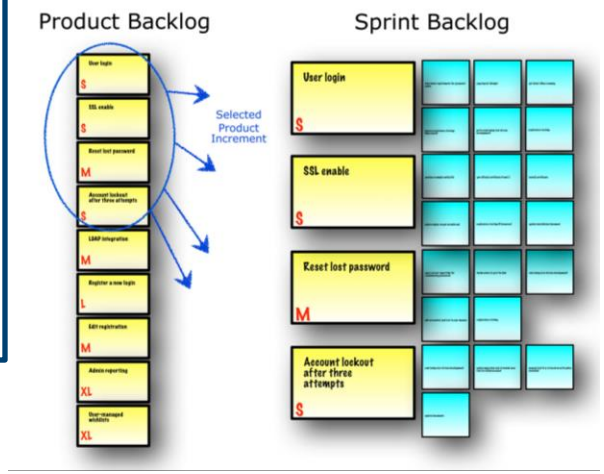


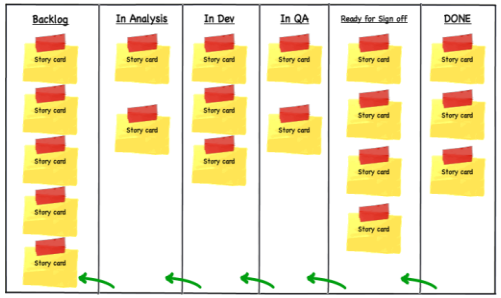
Scrum Way

Sprints are fixed length

Other stakeholders

Customer

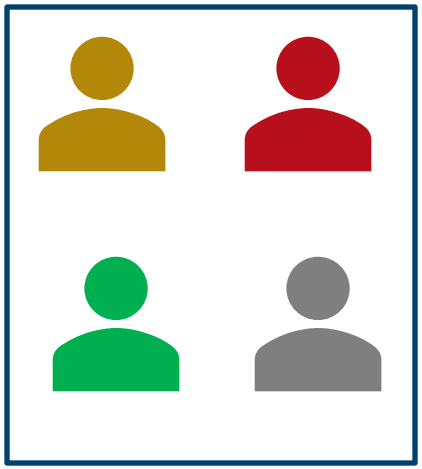




Not a manager

Product owner

Scrum master



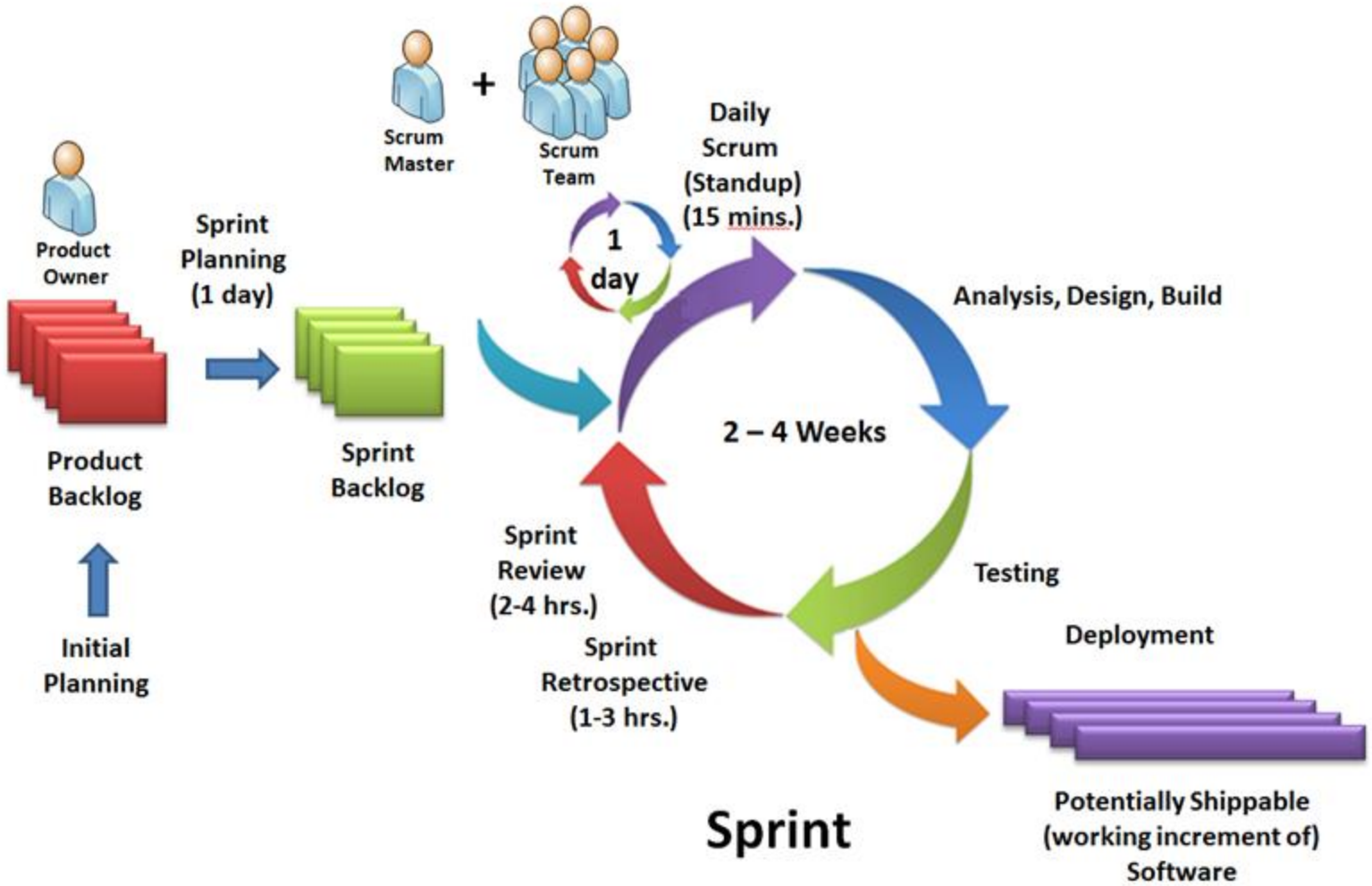
Cross-functional self-contained team

Daily Scrum Agenda

- Answer three questions at “daily scrums”:
  1. *What have you done since yesterday?*
  2. *What are you planning to do today?*
  3. *Are there any impediments or stumbling blocks?*

- Help individuals by identifying what they need

Scrum Sprint Cycle



Scrum Summary

- Self-organizing cross-functional small team with daily short standup meetings
- Work in sprints of 2-4 weeks
- For this course: Product Owner is member of the teaching team, Scrum Master changes with each sprint

Summary of Key Points

- A **software process** is the **set of activities** involved in producing and maintaining software

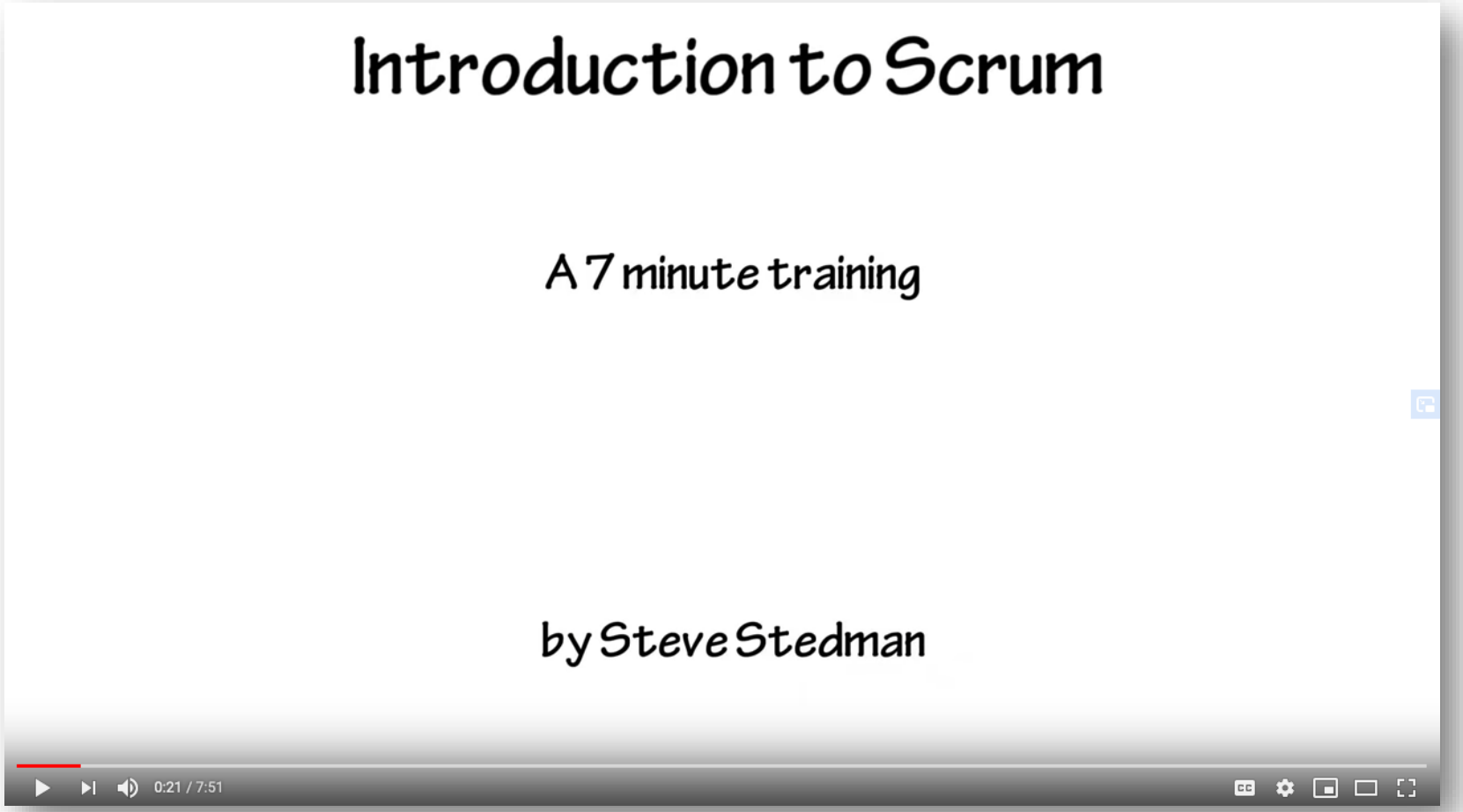
- All software processes include the **fundamental activities** of software specification, design, implementation, validation/verification, and maintenance

Summary of Key Points

- Software process models are **abstract representations** that describe the **organisation** of fundamental software process activities

- The choice of which process model(s) to use is dependent on the project **objectives and context**

Optional preparation for upcoming lectures



https://www.youtube.com/watch?v=9TycLR0TqFA