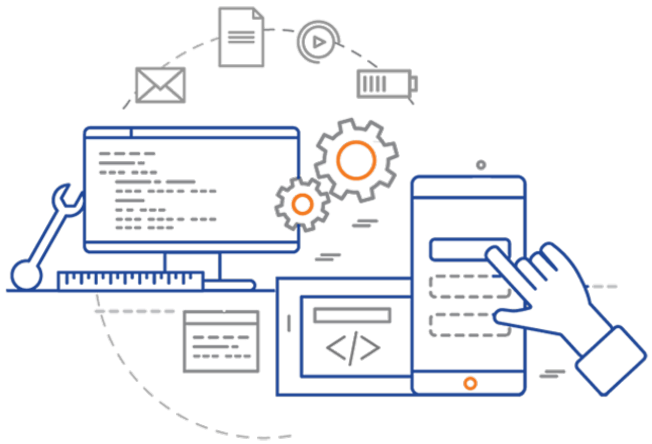


CS 398: Application Development

Week 01 Lecture: Software Process

Software process

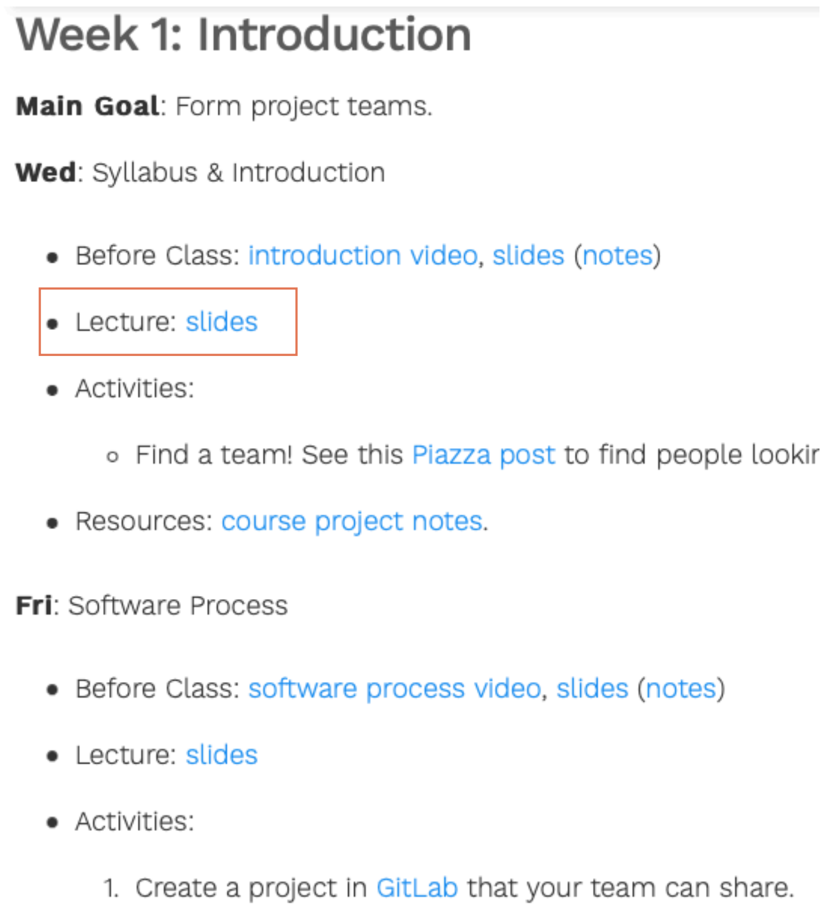
Logistics

Lecture slides

Terminology

- **Videos**: recorded lectures.
  - Videos and slides will be posted at the start of the week.

- **Lectures**: these online sessions.
  - Slides may be updated right up to the start of class.
  - I'll typically post them after the lecture.
  - I'll also record the lectures. They will be found directly in the MS Teams channel.



Week 1: Introduction

Main Goal: Form project teams.

Wed: Syllabus & Introduction

- Before Class: introduction video, slides (notes)
- Lecture: slides
- Activities:
  - Find a team! See this Piazza post to find people lookir
- Resources: course project notes.

Fri: Software Process

- Before Class: software process video, slides (notes)
- Lecture: slides
- Activities:
  1. Create a project in GitLab that your team can share.

Fri: Software Process

- Lectures: slides
- Videos: video, slides (notes)
- Activities: Create a project
    - Create one project in GitLab that your team can share.
        - Create under one person's account, and then add everyone else on the team as Owners/Developers (Members on the sidebar).
        - Add the instructor and the course staff so that they can view your work! They don't need developer access, just the ability to view content. Refer to the list of course staff for email addresses.
    - Email the instructor with a list of team members and link to the project.

**Today please!
I'd like to start Team Meetings on Monday.**

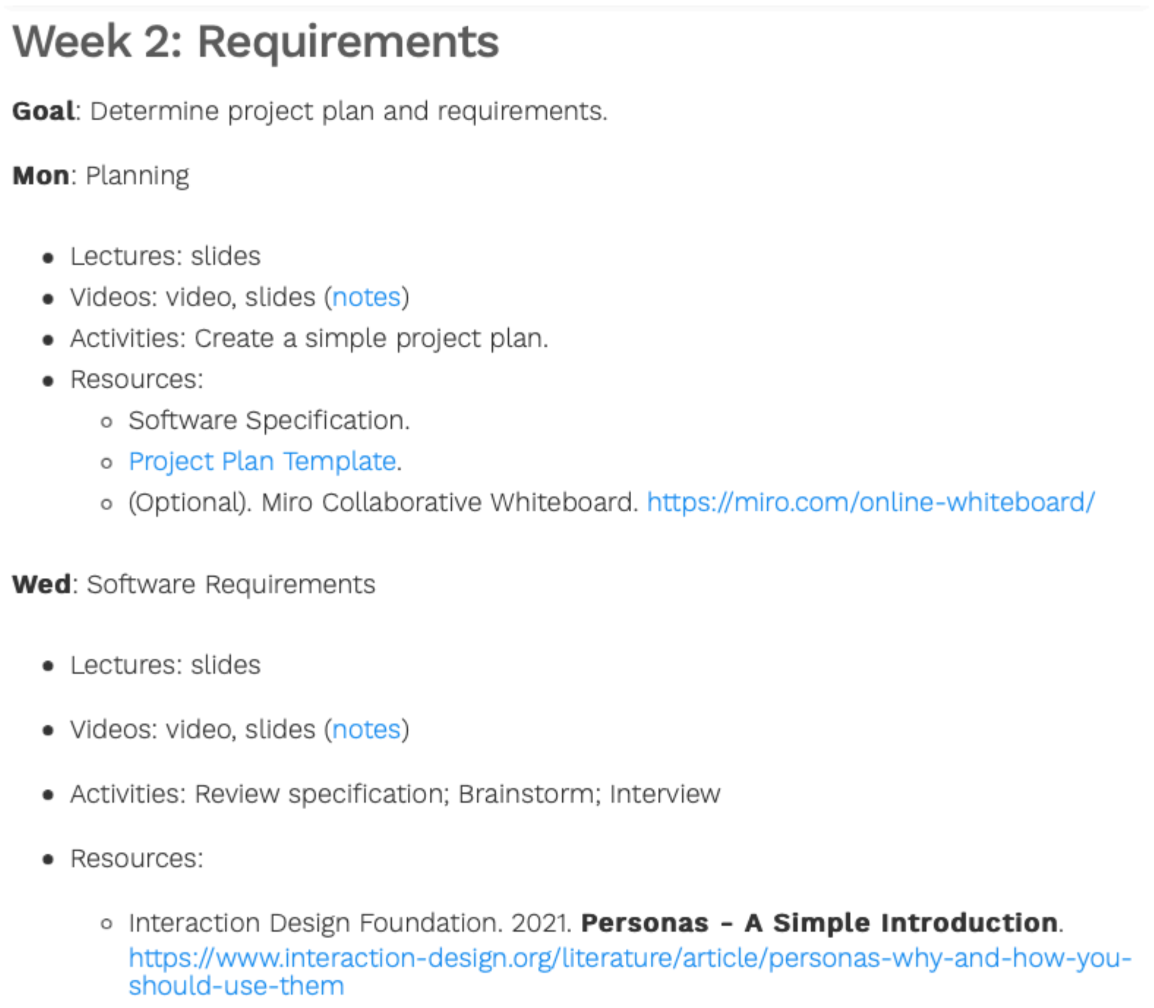
https://student.cs.uwaterloo.ca/~cs398/01-syllabus/1-weekly-schedule/#week-1-introduction

Next week!

- Product specification
- Project planning
- Requirements definition

Stretch Goal: I would like all project teams formed by the end of today so that I can setup project channels over the weekend!

i.e. Monday will (hopefully) be shortened lectures, and time to work on your project plan.



Week 2: Requirements

Goal: Determine project plan and requirements.

Mon: Planning

- Lectures: slides
- Videos: video, slides (notes)
- Activities: Create a simple project plan.
- Resources:
  - Software Specification.
  - Project Plan Template.
  - (Optional). Miro Collaborative Whiteboard. https://miro.com/online-whiteboard/

Wed: Software Requirements

- Lectures: slides
- Videos: video, slides (notes)
- Activities: Review specification; Brainstorm; Interview
- Resources:
  - Interaction Design Foundation. 2021. **Personas - A Simple Introduction**. https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them

Using GitLab

https://git.uwaterloo.ca

What is GitLab?

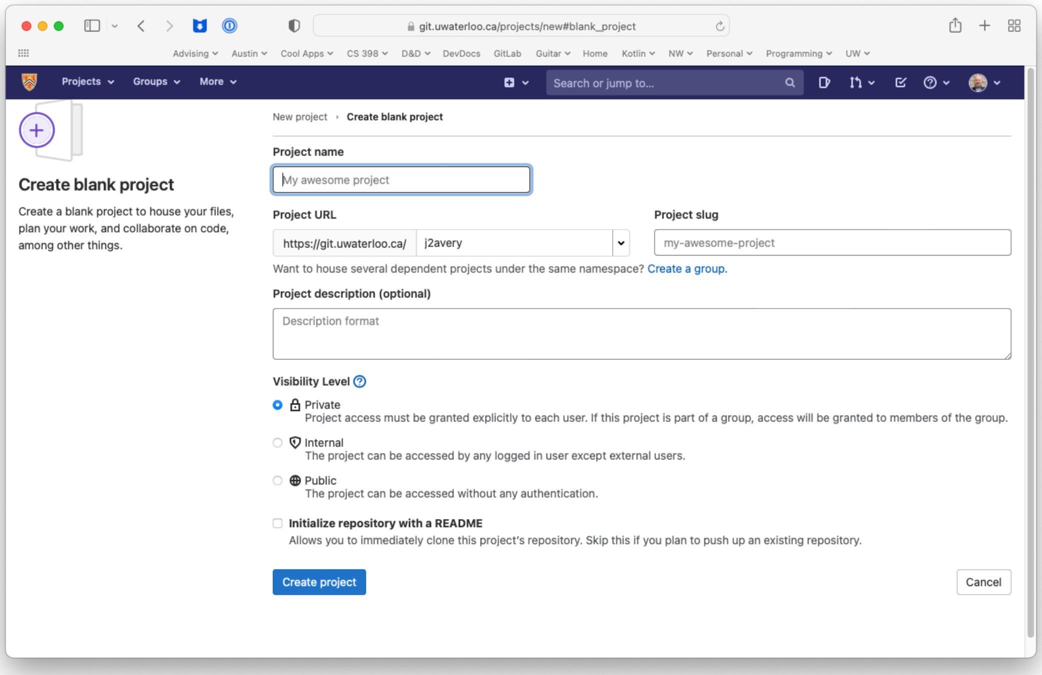
Feature	Description	
Repository	Version control for source code, or other files.	★
Issues	Mechanism to track project tasks or issues. They can capture details, be assigned to a milestone (e.g. Sprint 1), passed between people and have a status (e.g. Open, Closed).	★
Wiki	Create online documents using Markdown. This is very useful for longer-documents (e.g. design documents).	★
CI/CD	Continuous Integration and Deployment. We can setup rules that will automatically test or deploy our source code when it's committed (or other conditions are met).	
Snippets	Share snippets of source code outside of the project.	

Using GitLab

https://student.cs.uwaterloo.ca/~cs398/a-reference/1-gitlab/

To start using GitLab:

1. Create a new blank project in GitLab with a meaningful name and description.
2. When it opens, select `Members` from the left-hand menu. `Invite` each member of your team with the appropriate role: typically, `Developer` or `Maintainer` for full access.
3. Optional. Under `Settings – General`, add a project description. Create and upload an avatar!



New project › Create blank project

Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

Project name

My awesome project

Project URL

https://git.uwaterloo.ca/ j2avery

Project slug

my-awesome-project

Want to house several dependent projects under the same namespace? Create a group.

Project description (optional)

Description format

Visibility Level

- [✓ Private] Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.
- Internal — The project can be accessed by any logged in user except external users.
- Public — The project can be accessed without any authentication.

☐ **Initialize repository with a README**
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project · Cancel

Software Process

Process Activities

Example: building a house

These activities are common to any type of project - not just software.

- **Communication**: Discuss the goals and requirements with the customer, potential users, and other stakeholders. This is critical to ensure that we're solving the correct problem.
- **Planning**: Identify people and resources, potential risks to the project, and developing a plan to mitigate these risks. These are what we would typically call "project management" activities.
- **Modelling**: Design abstract models or representations of the product that you need to build.
- **Construction**: The process of building your product i.e. realizing your design. This may require successive iterations to complete, and should also include some level of testing and validation.
- **Deployment**: Tasks required to produce a working product, and deliver it to the customer. This includes collecting their ongoing feedback and making revisions as needed.

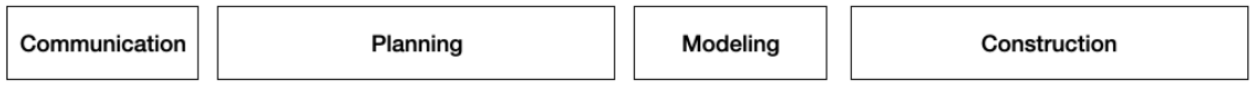
Discuss with a builder.

Hire a project manager.

Get blueprints!

The crew builds your house.

You inspect the results and take possession!



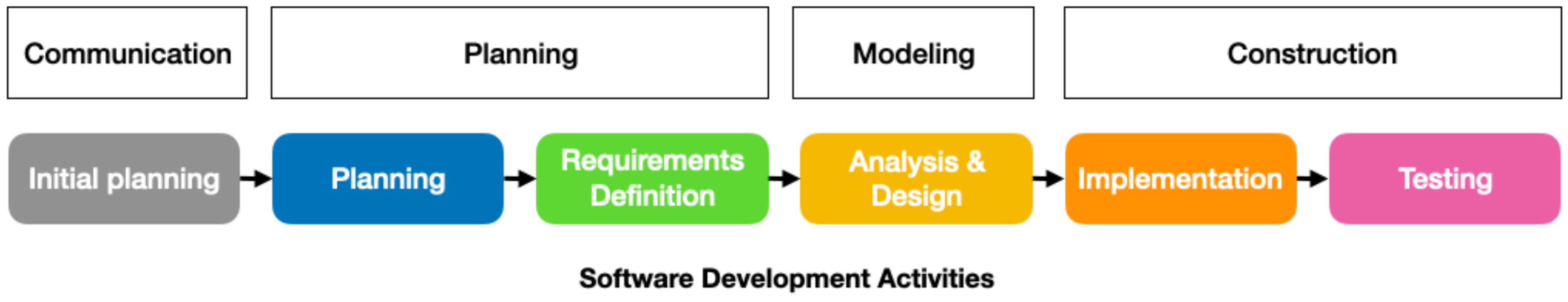
Do we do these in order? In parallel? How do they fit together?

Process Model?

We use the term **process model** to describe the structure that is given to these activities.

That is, it defines the complete set of activities that are required to specify, design, develop, test and deploy a system, and describes how they fit together. A software process model is a type of process model adapted to describe for software systems.

Below, we've listed the steps that we would typically consider as required for a software project.

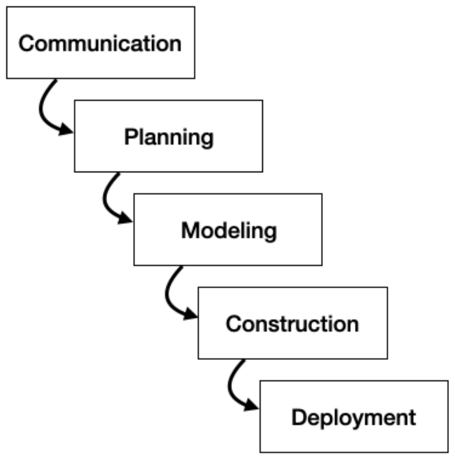


Stringing them all in a line is the world's simplest process model (assembly line?!)

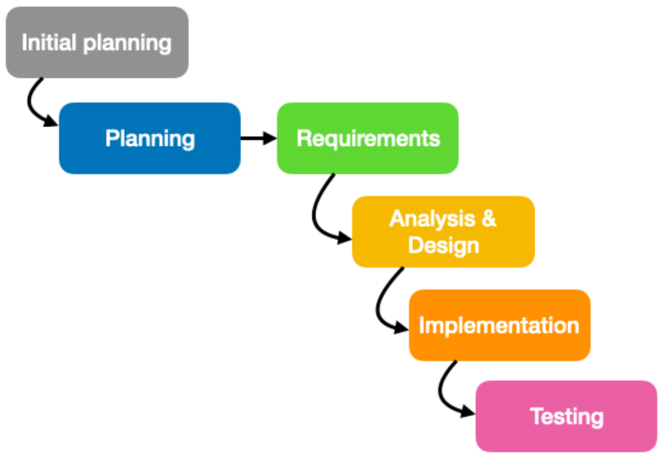
Waterfall Model

In a 1970 paper, Winston Royce laid out a mechanism for formalizing the large-scale management of software projects [Royce 1970], dubbed the Waterfall model. This envisions software production as a series of steps, each cascading into the next one, much like a waterfall.

Software development is treated as a set of linear steps that are followed strictly in-order.



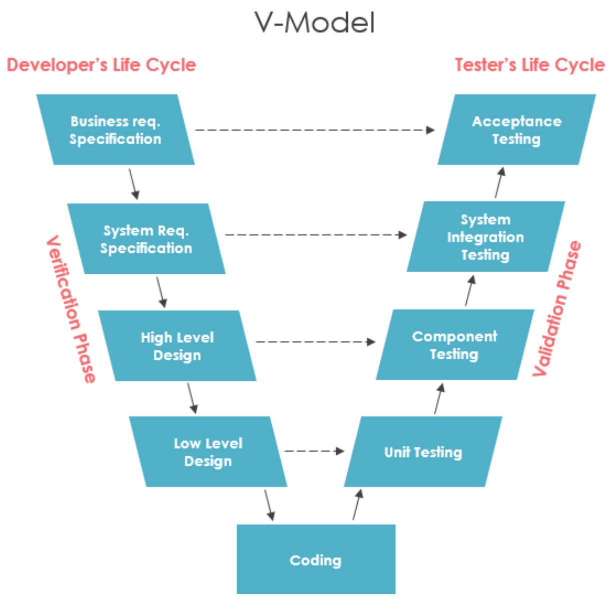
Linear Model (General)

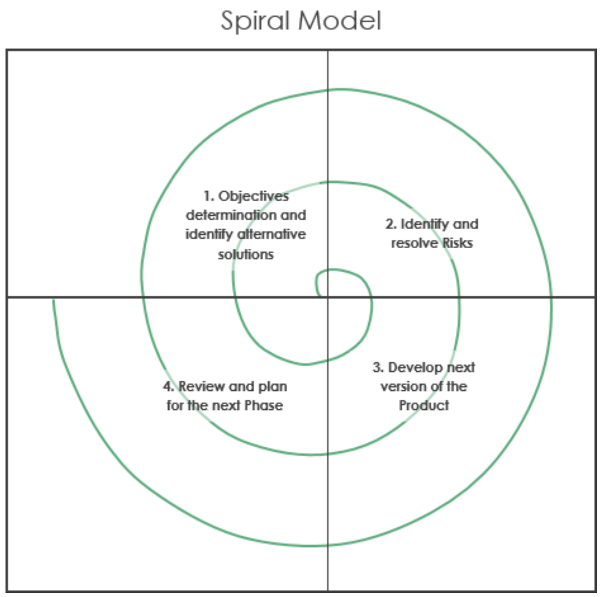


Waterfall Model (Software)

What is the problem with this approach?

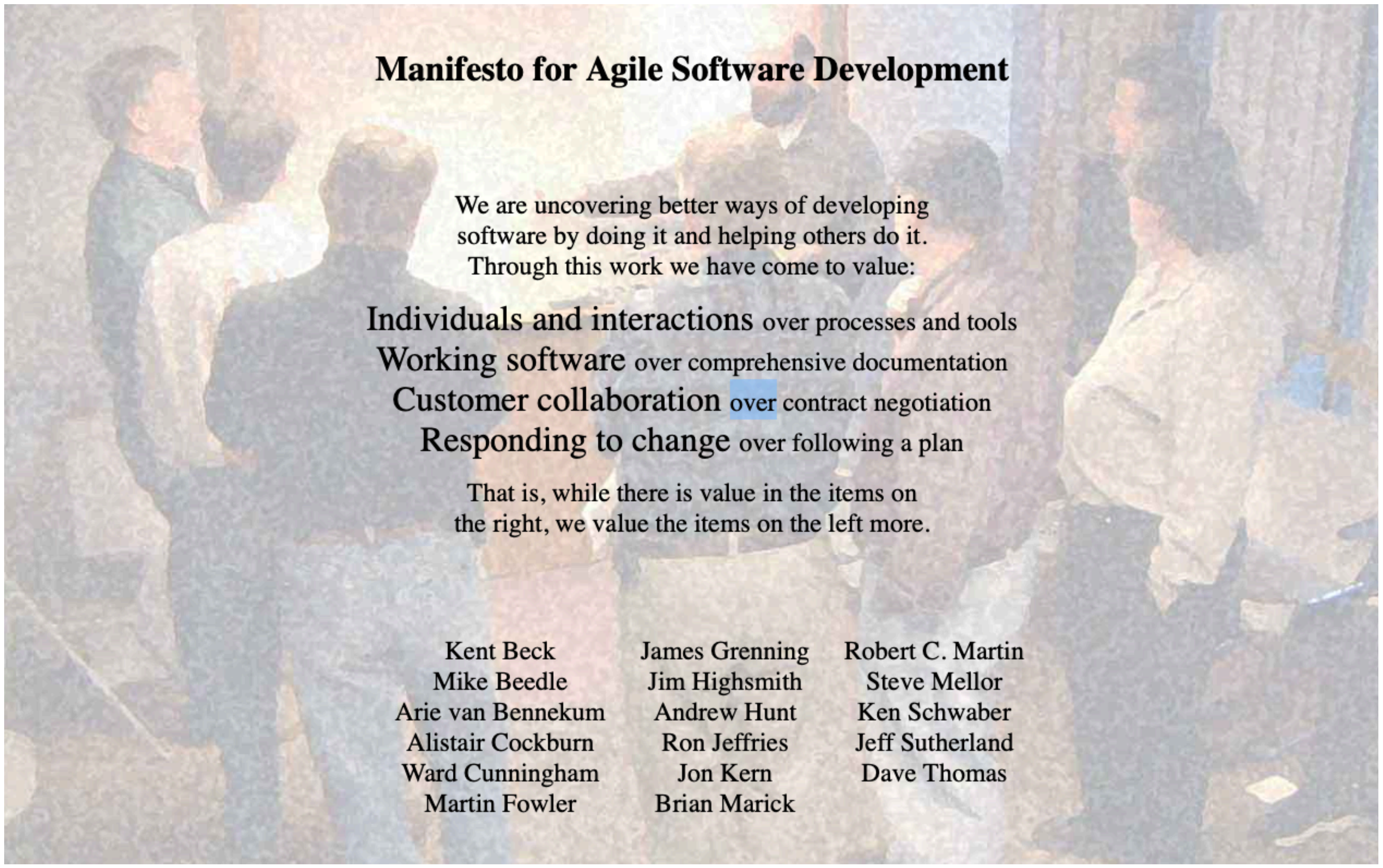
Other Process Models




Spiral Model
1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

V-Model lines up testing with the related area of interest. Acknowledges connections, but doesn't handle change very well.

Spiral model defines iterations, starting with abstract and progressing through levels of detail. Defined by product manager to direct product versions.



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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

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

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

https://agilemanifesto.org

What is Agile?

“Agile Software Development” isn’t a single process, but rather an approach to software development that encompasses this philosophy. It encourages **team structures and attitudes that make communication easier** (among team members, business people, and between software engineers and their managers). It emphasizes **rapid delivery of operational software**, but also recognizes that planning has its limits and that **a project plan must be flexible**. — Pressman & Maxim 2020.

Agility means recognizing that requirements and plans will change over time.

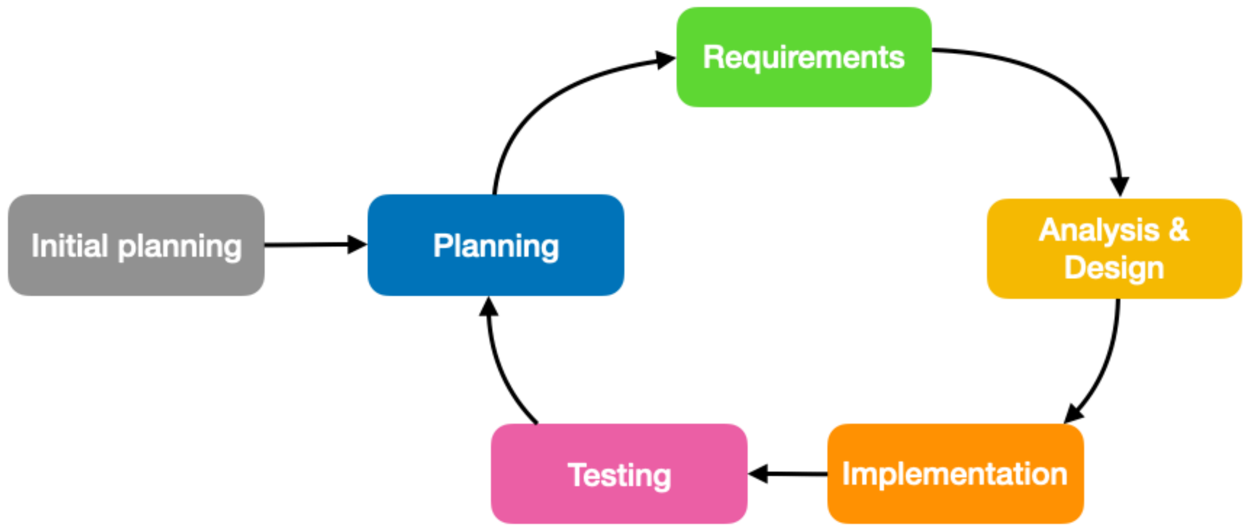
1. Software is too complex to design and build all at once. It’s more manageable to add features and test incrementally.
2. Requirements will change during the development cycle, and projects need to be responsive to this i.e. adaptable to change.

Challenges

1. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds.
2. For many types of software, design and construction are interwoven. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to predict how much design is necessary before construction is used to prove the design.
3. Analysis, design, construction, and testing are not as predictable as we might like (from a planning point of view).

How do we create a process that can manage this type of unpredictability?

We use an **iterative, evolutionary development model**. Instead of building a “complete” system and then asking for feedback, we instead attempt to deliver features in small increments, in a way that we can solicit feedback continuously though the process.



Iterative Software Model

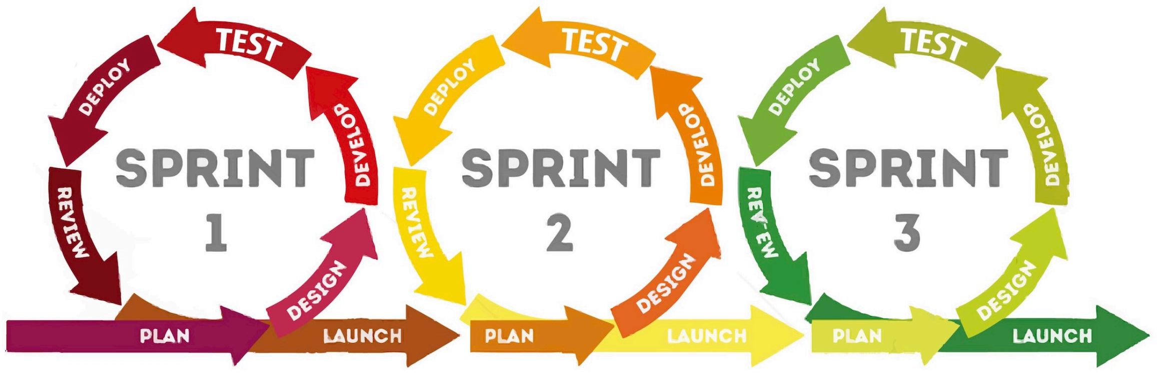
What about this? It’s much better, but it feels like we’re iterating too many activities.
e.g. do we really want to iterate over requirements every iteration?

Scrum

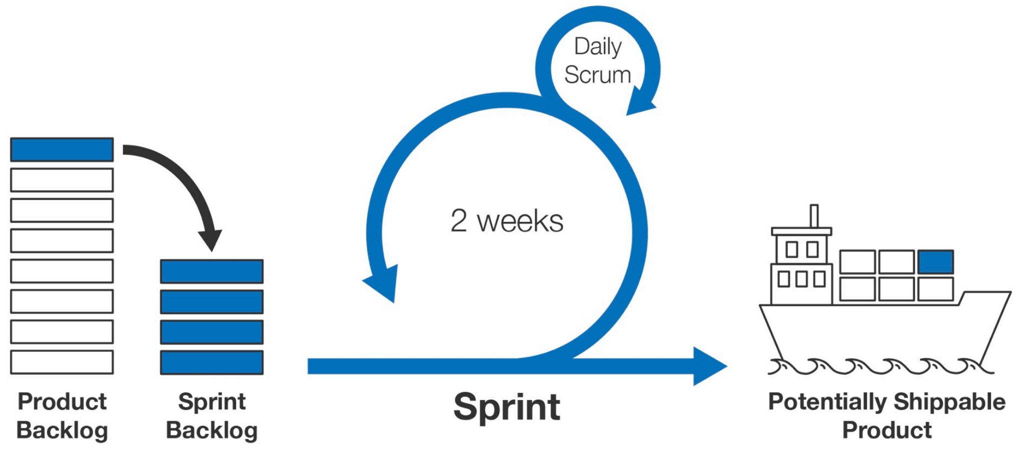
Scrum is one (of many) different Agile methods.

Scrum breaks down a project into fixed-length iterations called **sprints** (typically 2-4 weeks in length for each sprint). Sprints are defined so that you iterate on prioritized features in that time, and produce a fully-tested and shippable product at the end of each sprint.

Typically you will iterate until you and the customer together decide that you are "done".



Key Concepts



https://kunzleigh.com/about/our-approach/

Criticism: Not everything can be time boxed in 2-week chunks.

- **Product Backlog** is a list of all possible features and changes that could be considered. Collected from the customer's feedback, or ideas that the team has.
- **Product Owner** is the person responsible for gathering requirements and placing in the product backlog.
- **Sprint Backlog** is the set of features that are assigned to a sprint. This is the "scope" for that sprint.
- **Scrum Master** is the person that helps facilitate work during the sprint (like a "team lead").

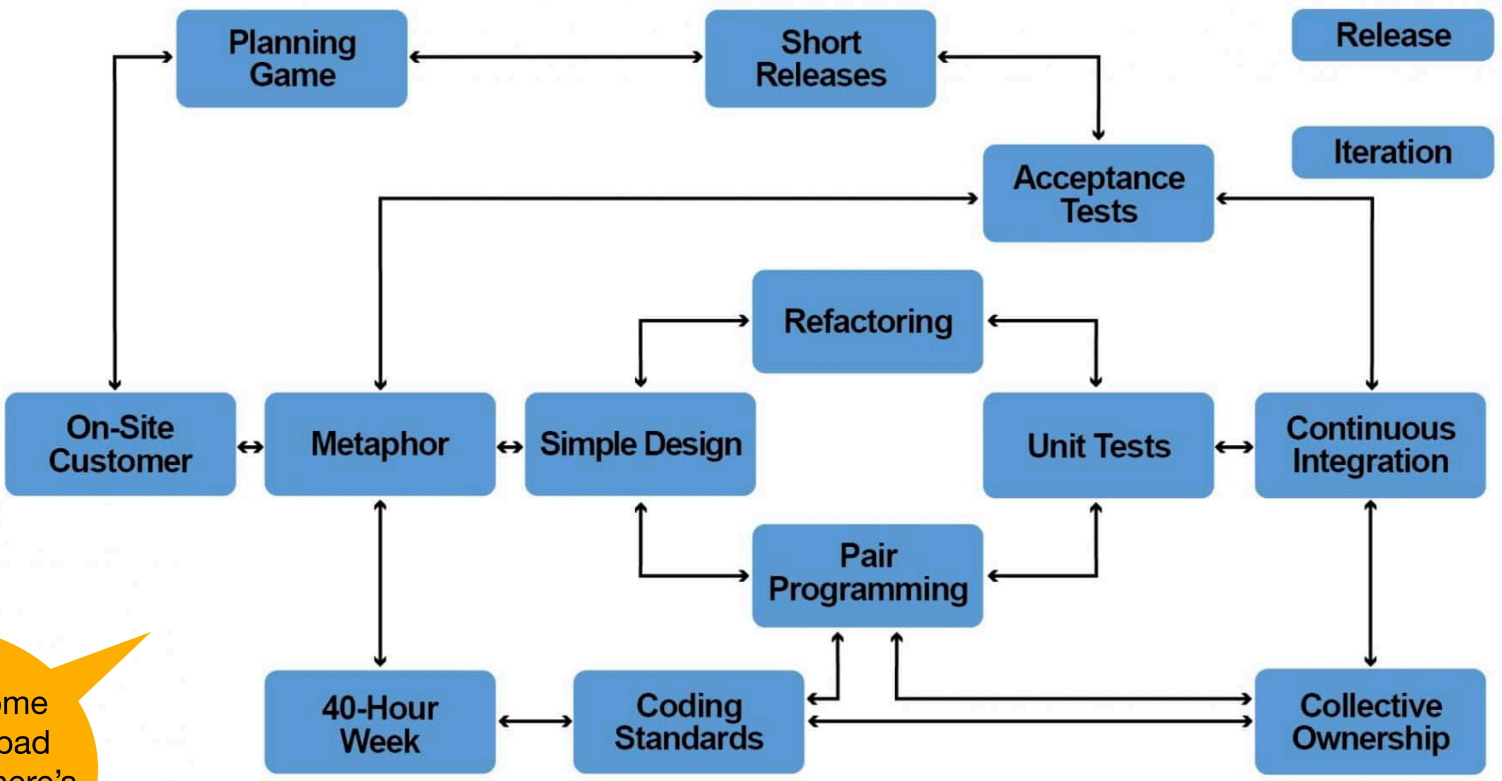
Extreme Programming

Extreme Programming (XP) is an Agile methodology focused on best-practices for programmers. It was based on a large-scale project that Kent Beck managed at Chrysler in the late 90s. It aims to produce higher-quality software and a higher quality-of-life for the development team.

The five core values of XP are:

- **Communication**: The key to a successful project. It includes both communication within the team, and with the customer. XP emphasizes face to face discussion with a white board (figurtively).
- **Simplicity**. Build the “simplest thing that will work”. Follow YAGNI (You Ain’t Gonna Need It) and DRY (Don’t Repeat Yourself).
- **Feedback**. Team members solicit and react to feedback right away to improve their practices and their product.
- **Courage**: The courage to insist on doing the “right thing”. The course to be honest with yourselves if something isn’t working, and fix it.
- **Respect**: Respect your team members, and remember that development is a collaborative exercise.

XP launched with 12 “best practices” of software development.



Criticism: Some of these are bad ideas. Also, where’s non-dev activities?

Agile Principles

From these different Agile models, we can extract a set of useful guiding principles [Pressman 2018]. This is what we aspire to do with our practices.

- **Principle 1**. **Be agile**. The basic tenets of agile development are to be flexible and adaptable in your approach, so that you can adjust if needed between iterations. Keep your technical approach as simple as possible, keep the work products you produce as concise as possible, and make decisions locally whenever possible.
- **Principle 2**. **Focus on quality at every step**. The focus of every process activity and action should be the quality of the work produced.
- **Principle 3**. **Be ready to adapt**. When necessary, adapt your approach to constraints imposed by the problem, the people, and the project itself.
- **Principle 4**. **Manage change**. The approach may be either formal or informal, but mechanisms must be established to manage the way changes are requested, assessed, approved, and implemented.
- **Principle 5**. **Build an effective team**. Software engineering process and practice are important, but the bottom line is people. Build a self-organizing team that has mutual trust and respect.
- **Principle 6**. **Establish mechanisms for communication and coordination**. Projects fail because important information falls into the cracks and/or stakeholders fail to coordinate their efforts to create a successful end product. Keep lines of communication open. When in doubt, ask questions!

Agile Process

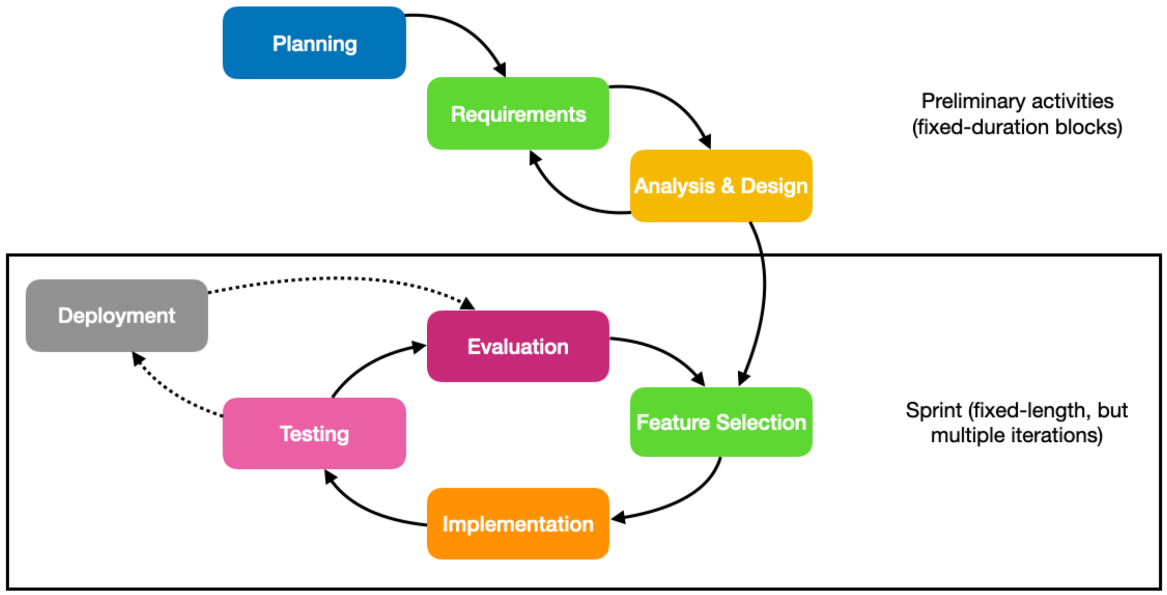
Preliminary Activities

Week 1: Now
Week 2: Planning, Requirements
Week 3: Architecture
Week 4: Design

Sprints

We have 4 sprints.
Each one is 2 weeks long.

Weeks 5-6: Sprint 1
Weeks 7-9: Sprint 2
Weeks 10-11: Sprint 3
Weeks 12-13: Sprint 4



Software Development Lifecycle (SDLC)

Sprint Breakdown

Week 1:
- Mon: Kickoff
- Wed/Fri: Work

Week 2:
- Mon/Wed: Work
- Fri: Demo

This addresses some key criticisms of SCRUM:

1. It recognizes that Requirements definition can take time, and you want to do *some* degree of analysis ahead of time.
2. Iteration between Requirements/A&D allows us to perform Design Iterations with users.
3. The iterations at the bottom are functionally equivalent to a sprint.

Agile Practices

Being Agile includes both the process AND some best practices that help us meet our goals.

We will discuss these in upcoming sections, in relation to the phase where they apply.

- **User stories**: describe features in a way that makes sense to customers.
- **Test-driven development**: tests are written before the code. This helps to enforce contracts/ interfaces.
- **Refactoring**: small continual improvements to code should be done routinely.
- **Pair programming**: critical code is written by two people working as a team.
- **Code reviews**: code changes need to be reviewed by one or more other developers on the team.