CASE WESTERN RESERVE UNIVERSITY

Agile Methods

CSDS 393/493: Software Engineering

Spring 2025

Agenda

- Agile practices
  - Manifesto
  - XP
  - Scrum
- Agile project management
- Scaling agile

Team Projects

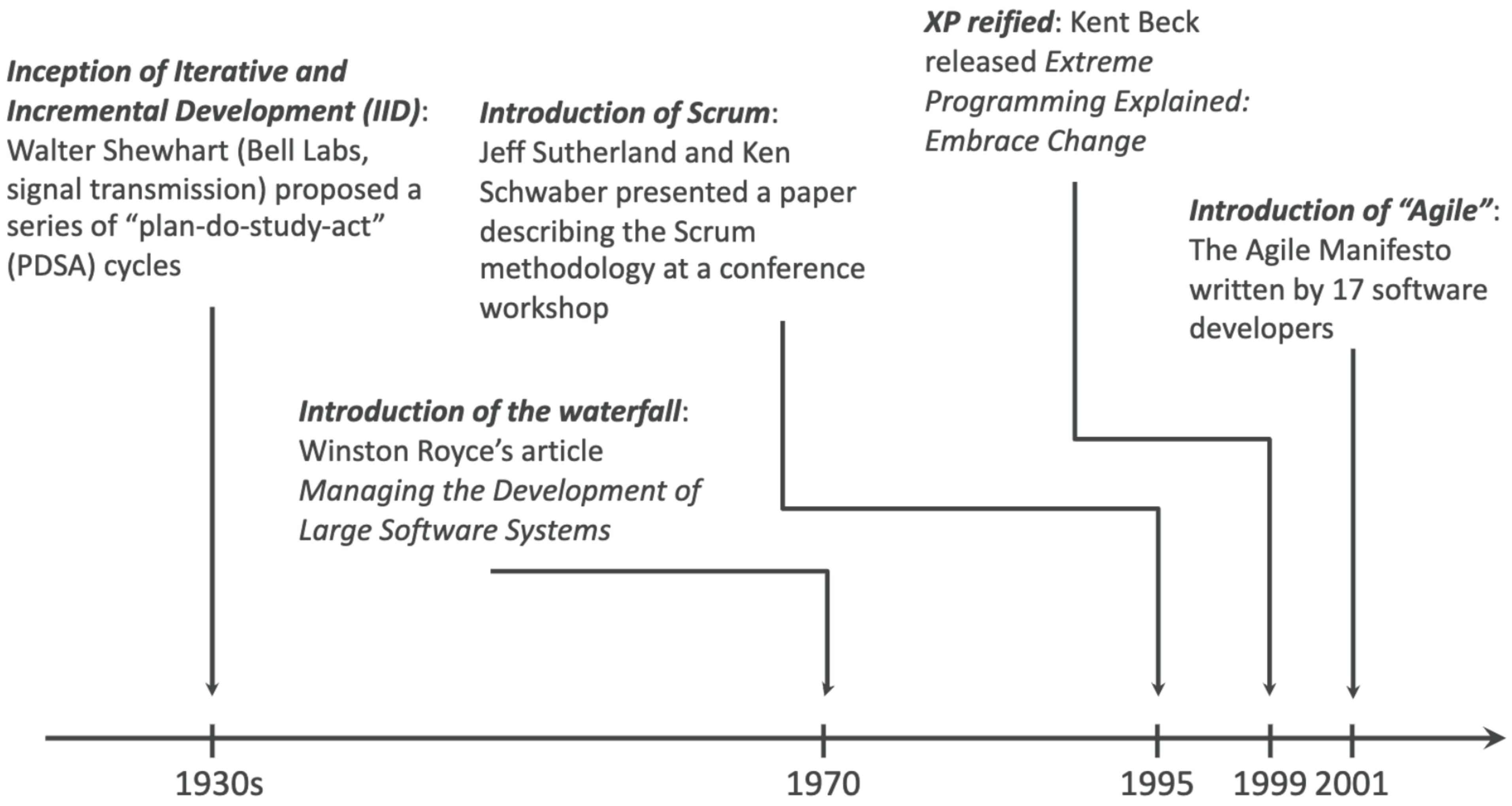
- Project Proposal assignment due today
- Weekly meeting with the TA
  - Ice-breaking
  - Make the project idea strong
- Project milestones:
  - **Demo 1**: Week of Feb 11
  - **Demo 2**: Week of Mar 4
  - **Demo 3**: Week of April 1
  - **Project presentation**: Week of April 15

Agile Software Development Is …

Both:

- a set of software engineering best **practices** (allowing for rapid delivery of high quality software)
- a **business approach** (aligning development with customer needs and goals)

Brief History of Agile



Inception of Iterative and Incremental Development (IID): Walter Shewhart (Bell Labs, signal transmission) proposed a series of "plan-do-study-act" (PDSA) cycles

Introduction of the waterfall: Winston Royce's article *Managing the Development of Large Software Systems*

Introduction of Scrum: Jeff Sutherland and Ken Schwaber presented a paper describing the Scrum methodology at a conference workshop

XP reified: Kent Beck released *Extreme Programming Explained: Embrace Change*

Introduction of "Agile": The Agile Manifesto written by 17 software developers

Two Major Influence in 1990s

- Object-oriented programming replaced procedural programming

- Rise of the Internet emphasized speed-to-market and company growth

Agile in a Nutshell

- A project management approach that seeks to respond to change and unpredictability,
  - primarily using incremental, iterative work sequences (often called “sprints”).

- Also: a collection of practices to facility that approach.
- Principles outlined in “The Manifesto for Agile Software Development.”

The Manifesto for Agile Software Development

Beck et al., 2001

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

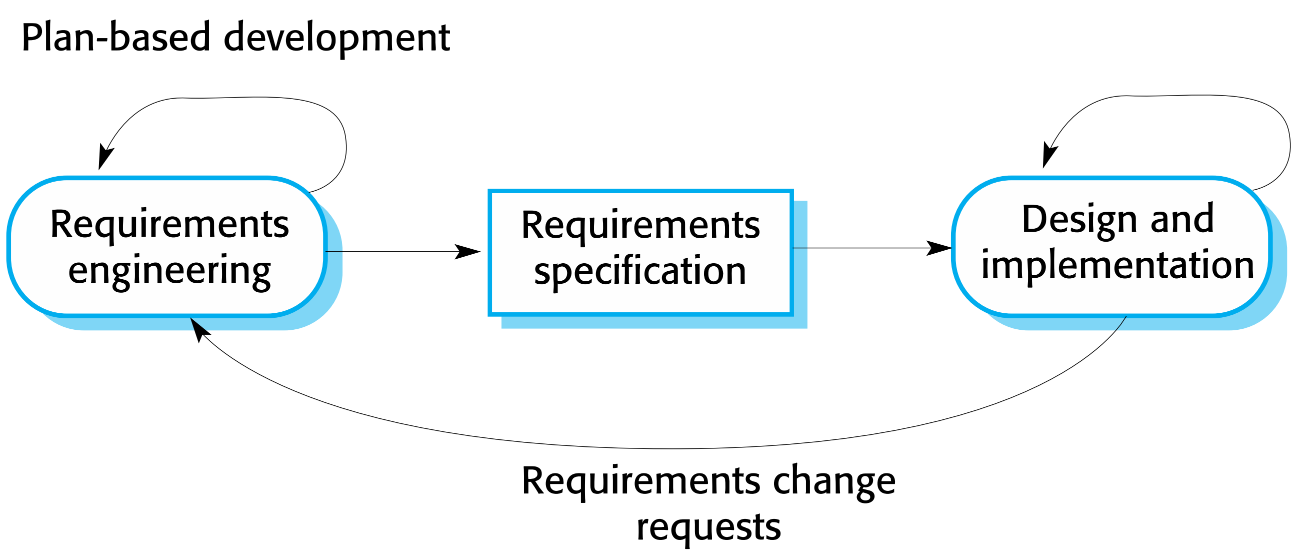
Programming is 4 Activities

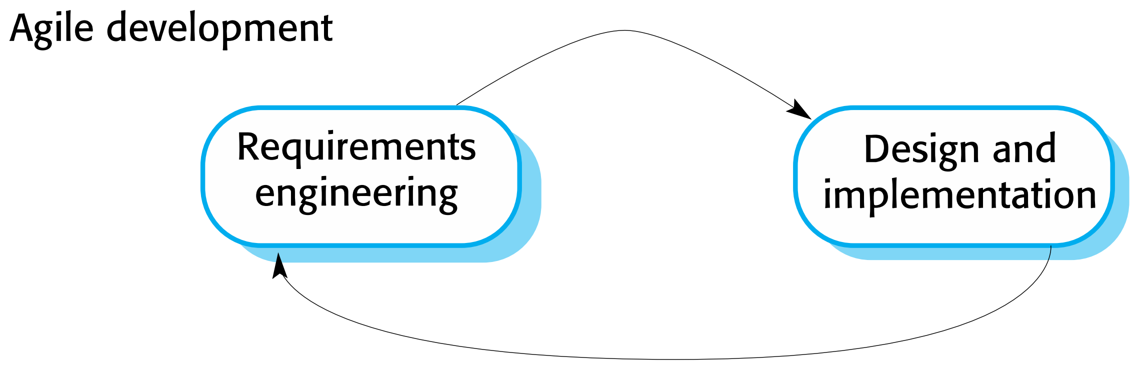
"**Listening, Testing, Coding, Designing**.

That's all there is to software. Anyone who tells you different is selling something."

–Kent Beck (Extreme Programming Explained)

Plan-driven and Agile Development


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests



Extreme Programming (XP)

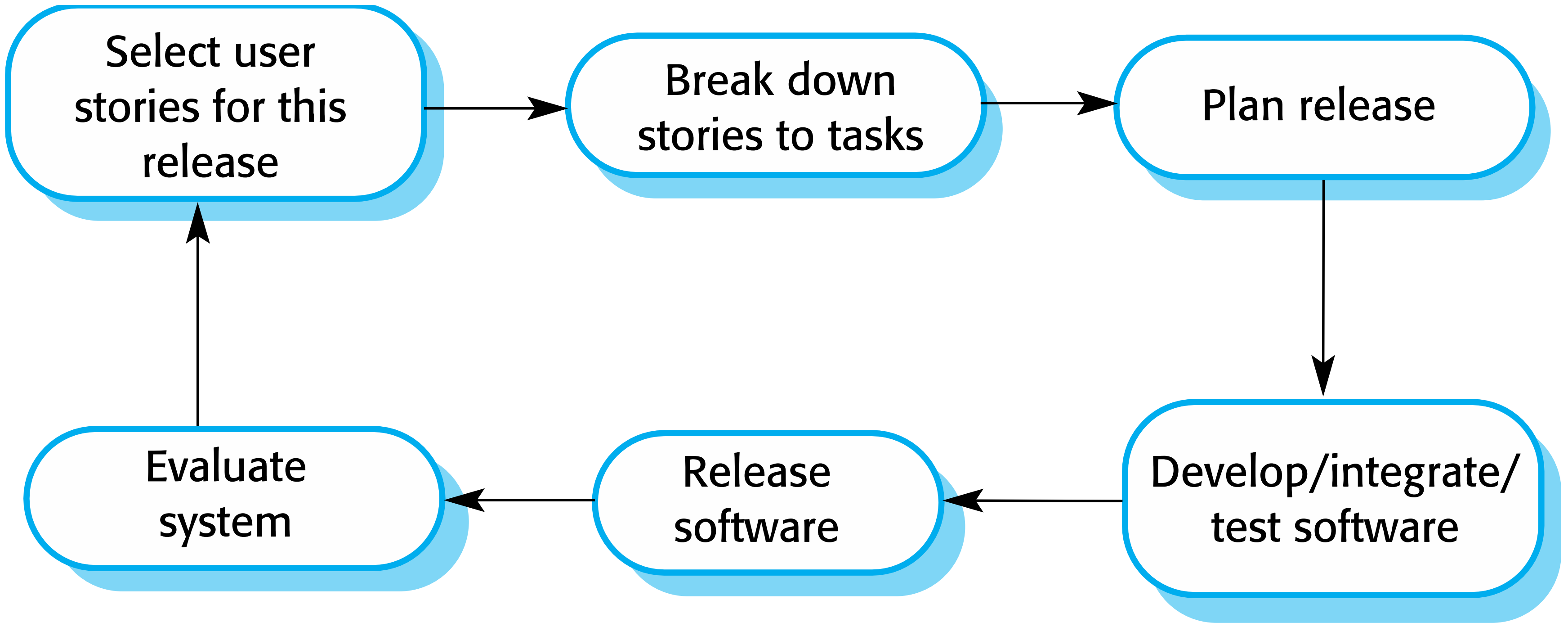
- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques.

- XP takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day
  - All tests must be run for every build and the build is only accepted if tests run successfully

XP Principles

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

The XP Release Cycle


Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

XP Practices

- Metaphor: "The system metaphor is a story that everyone can tell about how the system works"
  - Desktop Metaphor
  - Spreadsheet Metaphor
  - Shopping Cart Metaphor
  - Auction Metaphor
  - Blackboard Metaphor

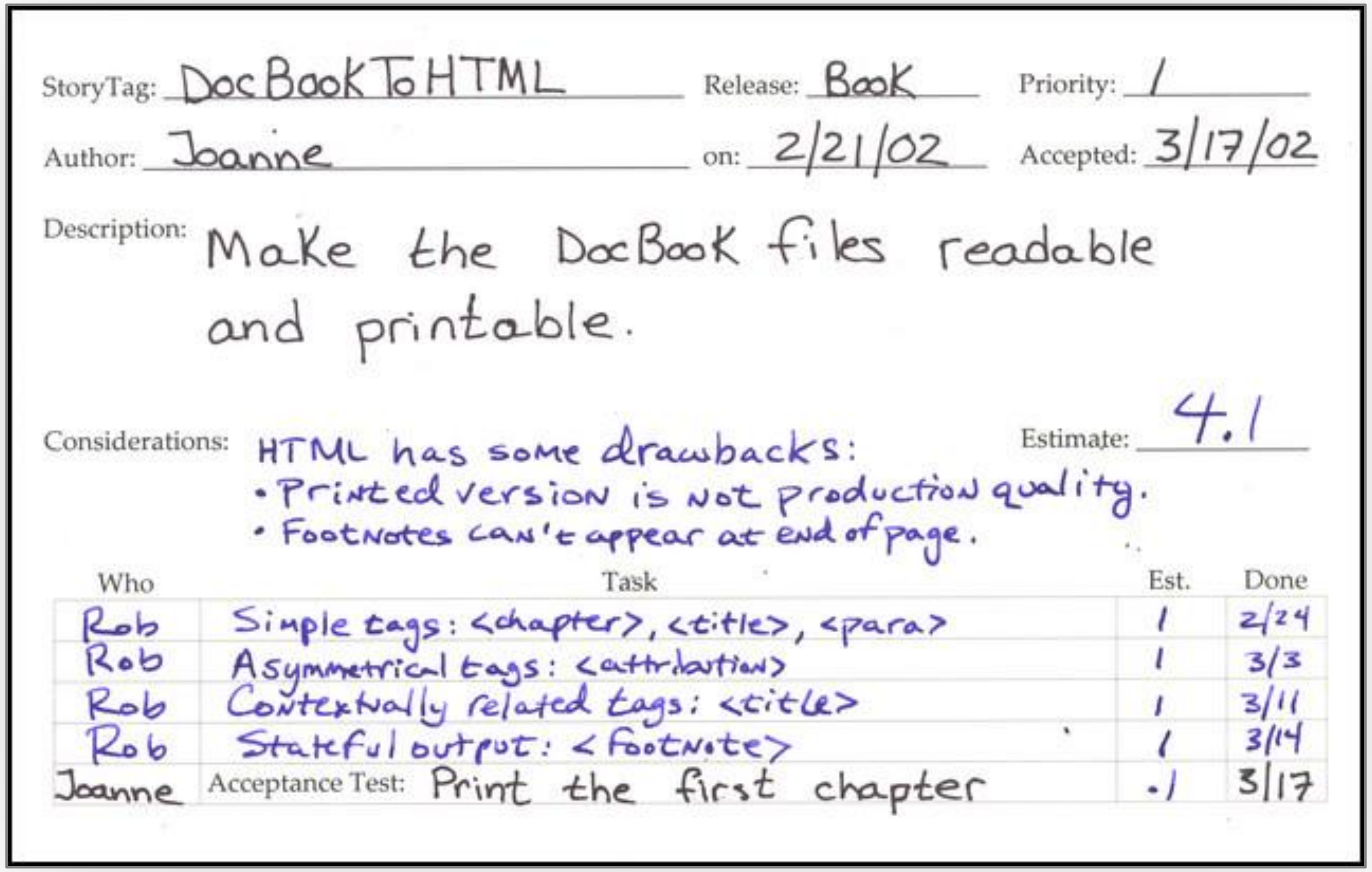
- The Planning Game:
  - Customers, managers, and developers meet to flesh out, estimate, and prioritize requirements for the next release.
  - The requirements are called "user stories" and are captured on "story cards".

Example User Stories

- Students can purchase monthly parking passes online.
- Parking passes can be paid via credit cards.
- Parking passes can be paid via PayPal.
- Professors can input student marks.
- Students can obtain their current seminar schedule.
- Students can order official transcripts.
- Students can only enroll in seminars for which they have prerequisites.
- Transcripts will be available online via a standard browser.

https://agilemodeling.com/artifacts/userStory.htm

Example User Story Card



StoryTag: DocBookToHTML   Release: Book   Priority: 1

Author: Joanne   on: 2/21/02   Accepted: 3/17/02

Description: Make the DocBook files readable and printable.

Considerations: HTML has some drawbacks:
- Printed version is not production quality.
- Footnotes can't appear at end of page.

Estimate: 4.1

Who	Task	Est.	Done
Rob	Simple tags: <chapter>, <title>, <para>	1	2/24
Rob	Asymmetrical tags: <attribution>	1	3/3
Rob	Contextually related tags: <title>	1	3/11
Rob	Stateful output: <footnote>	1	3/14
Joanne	Acceptance Test: Print the first chapter	.1	3/17

Examples of Task Cards for Prescribing

Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

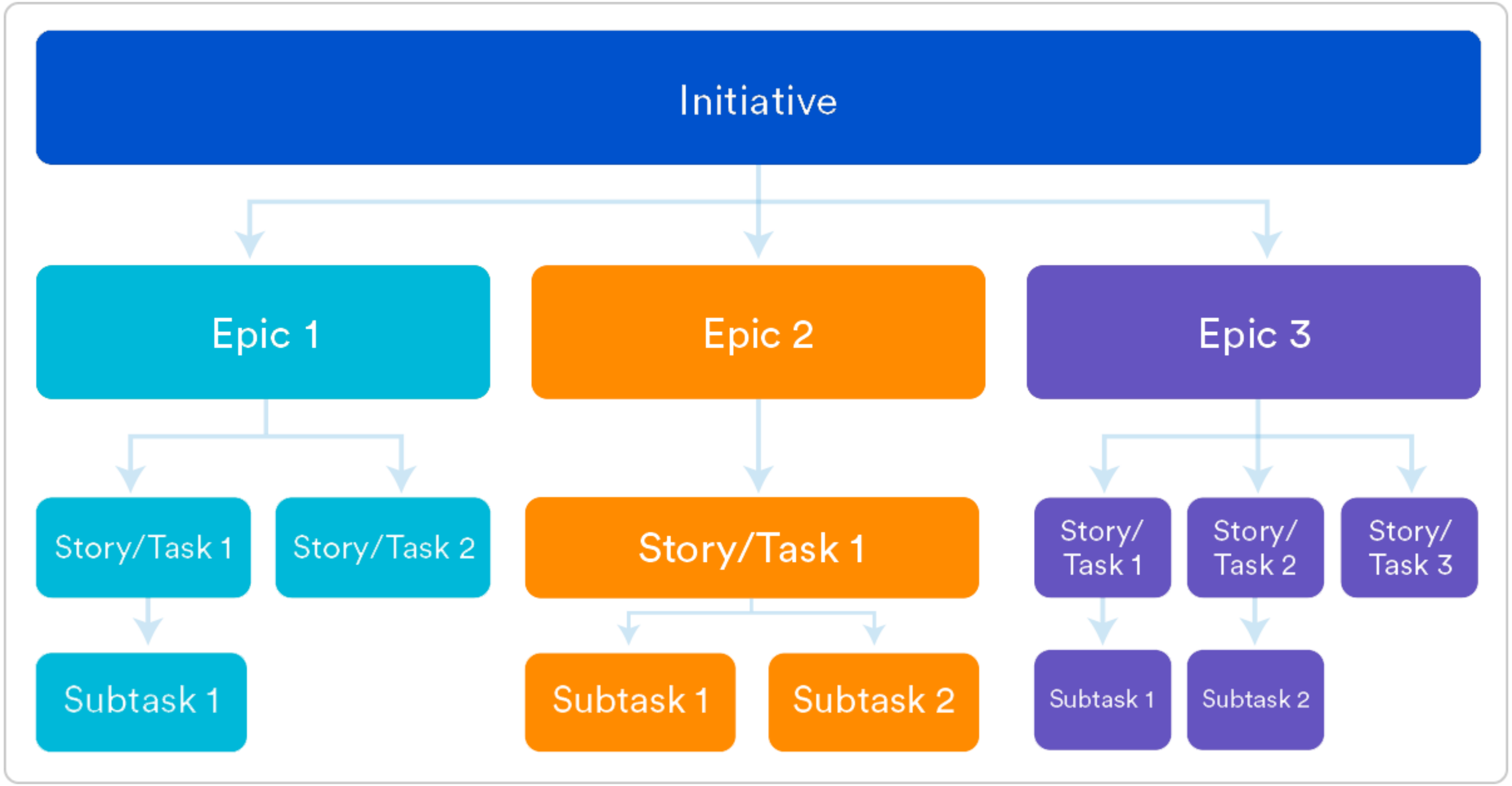
Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

Detailing a User Story

- A user story must be fleshed out, e.g., during
  - Analysis and modeling with stakeholders
  - Iteration planning
  - Implementation

- An epic is a large user story that can be broken down into smaller ones.

Story Structure



https://www.atlassian.com/agile/project-management/epics-stories-themes

XP Practices: Simple Design

- The system should be designed as simply as possible at any given moment.
- Extra complexity is removed as soon as it is discovered.

Test-Driven Development

- Developers write unit tests before they write the code itself. They are run when the code is written or changed.
- Customers write functional tests for each iteration, and at the end of each iteration all tests should be run.
- All code must pass all tests before it can be released.

Test Case Description for Dose Checking

Test 4: Dose checking

Input:

1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:

1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:

OK or error message indicating that the dose is outside the safe range.

Problems with Test-first Development

- Programmers prefer programming to testing
- Some tests can be very difficult to write incrementally
- It difficult to judge the completeness of a set of tests

Open Workspace

Small Releases

- An initial version of the system is put into production after the first few iterations.

- Subsequently, working versions are put into production anywhere from every few days to every few weeks.

Continuous Integration (CI)

Developers integrate new code frequently.

- Ideally the system is built after every commit.
- All automated tests must pass after integration or the new code is removed from the build

Refactoring

The design of the system is evolved through transformations of the existing design that keep all the tests running.

Programmers restructure (refactor) the system to

- Simplify
- Remove duplication
- Improve understandability
- Add flexibility

Examples of Refactoring

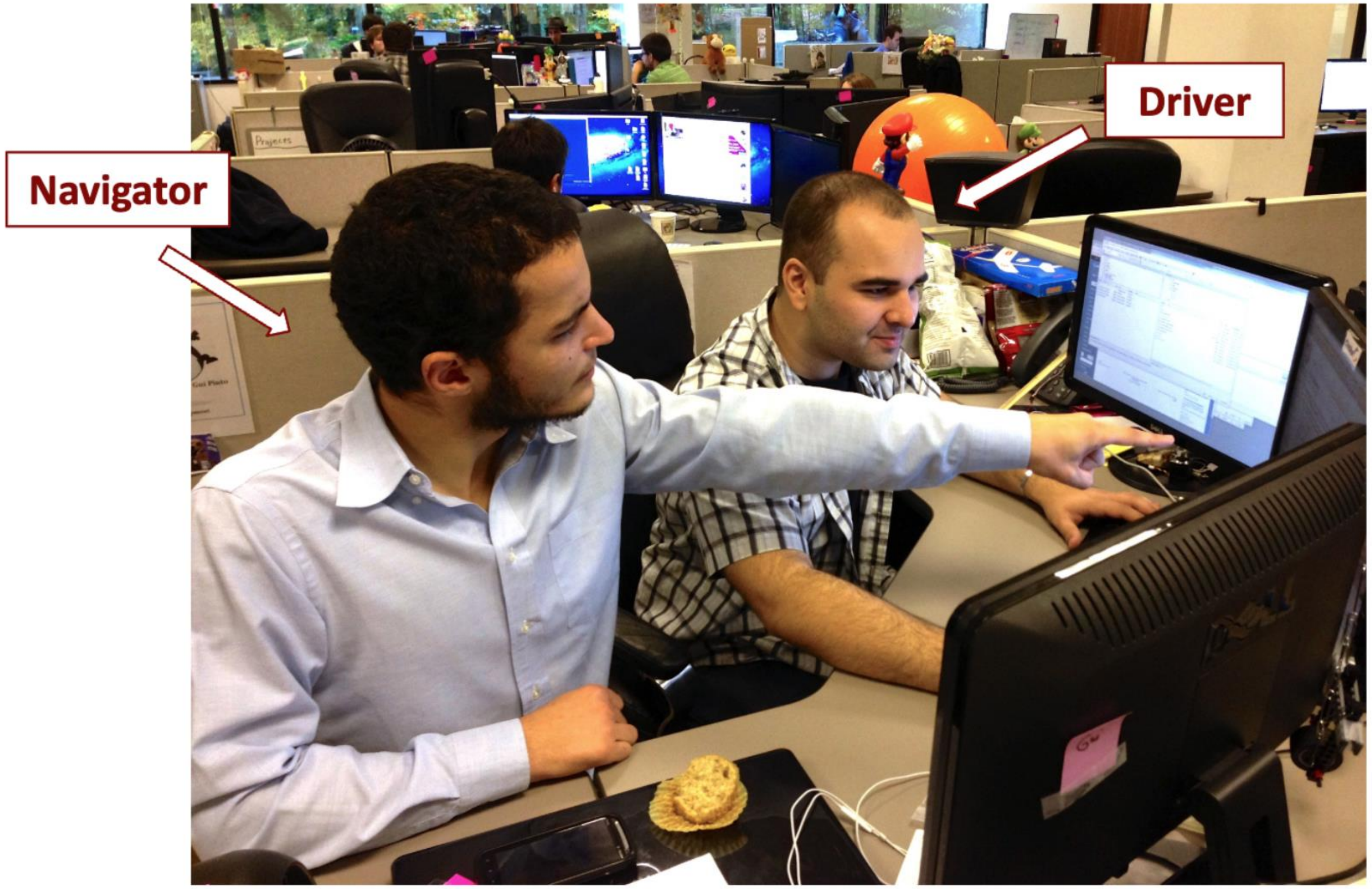
- Re-organization of a class hierarchy to remove duplicate code.
- Tidying up and renaming attributes and methods to make them easier to understand.
- The replacement of inline code with calls to methods that have been included in a program library.

Coding Standards

Programmers write all code in accordance with rules emphasizing communication through code.

- e.g., GNU coding standards
- Have you ever followed a coding standard?

Pair Programming



Collective Ownership

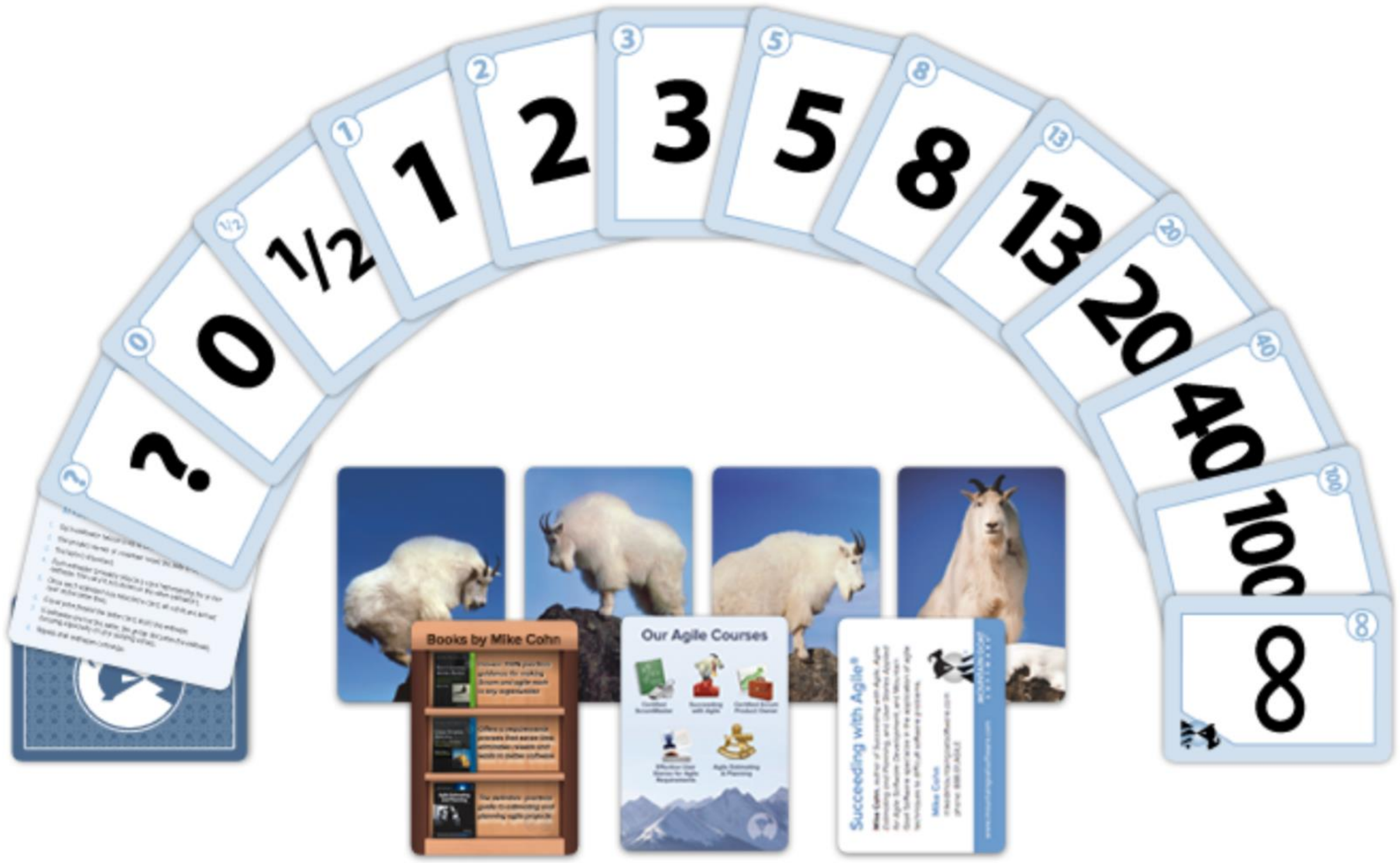
The code is owned by all developers, and they may make changes anywhere in the code at anytime they feel necessary.

- Risk?

40-Hour Week

No one can work a second consecutive week of overtime. Even isolated overtime used too frequently is a sign of deeper problems that must be addressed.

Planning Poker



On-Site Customer

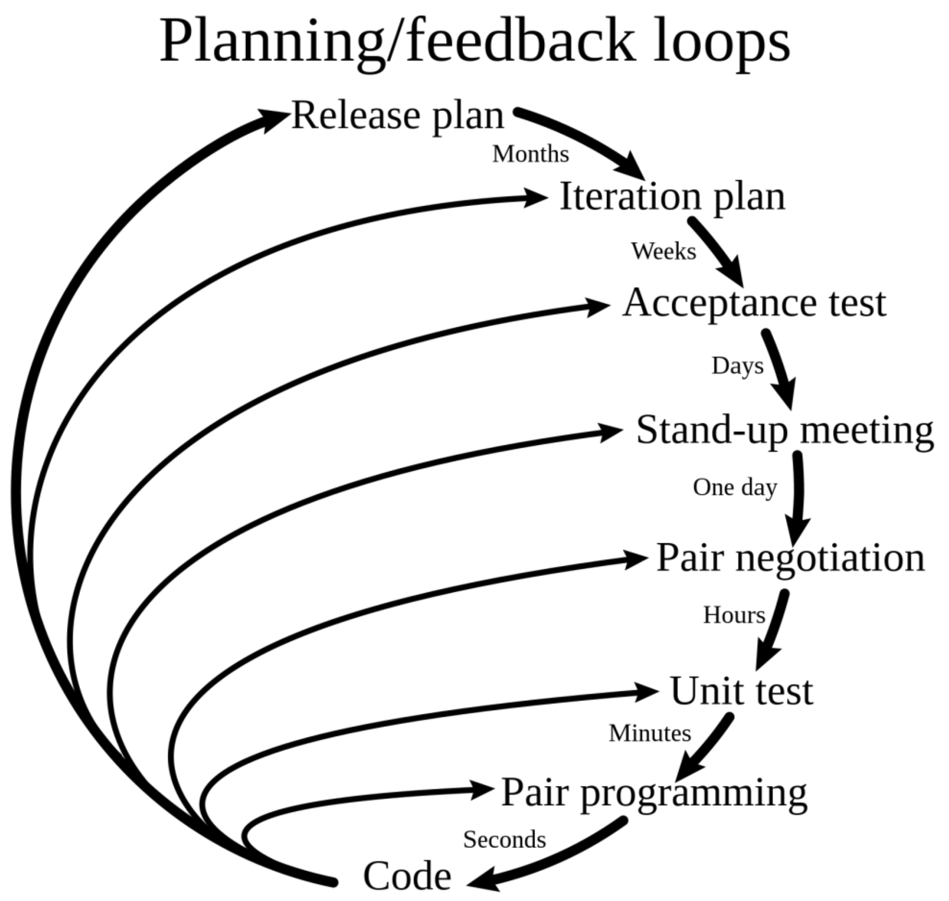
A customer works with the development team to

- answer questions
- perform acceptance tests
- ensure that develop is progressing as expected

XP Practices

Question: Which XP practices compensate for omitted conventional practices?

Planning and Feedback in XP



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

Question

Do you see any problems with XP?

Objections Raised to XP

- You need to do all of XP or none at all.
- XP is aimed at customers that don’t know what they want.
- Up-front analysis using mockups, storyboards, prototypes is less costly
- Constant refactoring entails high overhead and tends to introduce bugs.
- XP is over-reliant on testing.
- Many programmers dislike pair programming.
- The on-site customer representative is likely to be inexperienced.
- XP essentially requires a customer to sign an optional-scope contract.
- XP makes it hard to develop good early estimates of the work effort

Scrum

- Based on iterative and incremental development
- Somewhat more conventional than XP
- Projects split into iterations called sprints
- A sprint typically takes 1-4 weeks