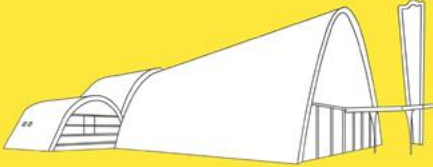


SOFTWARE ENGINEERING

A Modern Approach



MARCO TULLIO VALENTE

Chapter 2 - Processes

Prof. Marco Tulio Valente

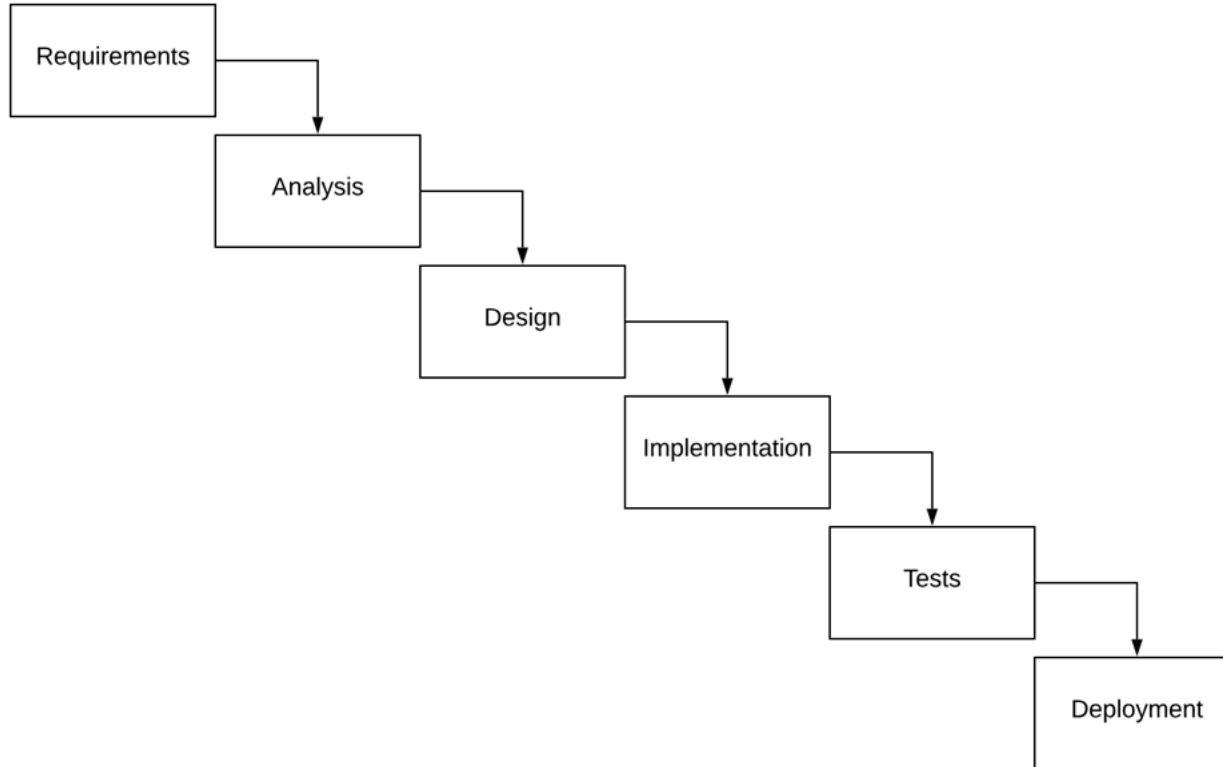
<https://softengbook.org>

CC-BY: This license enables anyone to distribute, remix, adapt, and build upon the material in any medium or format. The only condition is that attribution must be given to the author.

Traditional Engineering

- Civil, mechanical, electrical, aviation, automotive, etc
- Exists for thousands of years
- Two key characteristics:
 - Big Design Upfront (BDUF)
 - Sequential (Waterfall)

Thus, SE also started using Waterfall



But Waterfall did not work with software

Software is different

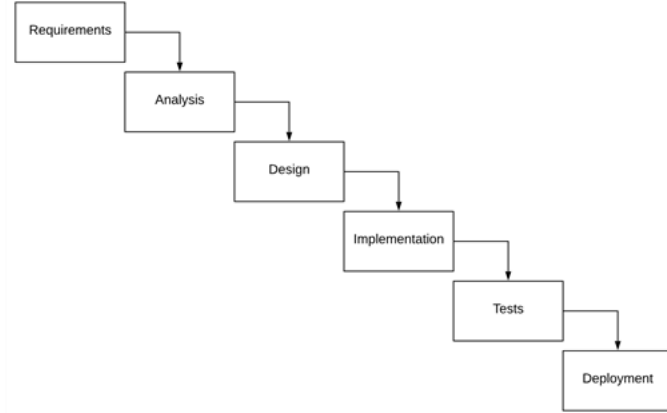
- Software Engineering \neq Traditional Engineering
- Software \neq (cars, bridges, houses, airplanes, phones, etc)
- Software \neq (physical products)
- Software is abstract and flexible

Challenge #1: Requirements

- Often, customers don't know what they want:
 - Feature space is “infinite” or hard to predict
 - World changes!

Challenge #1: Requirements

- It's not possible anymore to stay:
 - 1 year defining the requirements
 - 1 year designing the system
 - 1 year implementing the system
 - etc
- When the software is ready, it may already be obsolete!



Challenge #2: Detailed Documentation

- Verbose and of limited use
- In practice, not used during the implementation phase
- Plan-and-document did not work with software



Agile Manifesto (2001)

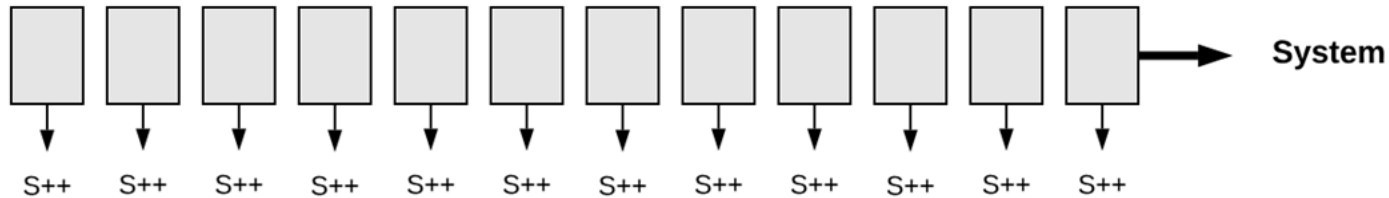


Key idea: iterative development

Waterfall



Agile



Iterative Development

- Let's consider a large and complex system
- What's the smallest feature increment we can deliver in 15 days and validate with users?
- Validation is very important
- Customers usually don't know what they want!

Other characteristics

- Less emphasis on documentation
- Less emphasis on big design upfront (BDUF)
- Customer involvement
- New programming practices: tests, refactoring, CI/CD, etc.

Agile Methods

Agile Methods

- Give consistency to agile ideas
 - Define a process, even if lightweight
 - Workflow, events, roles, practices, principles, etc

Agile Methods

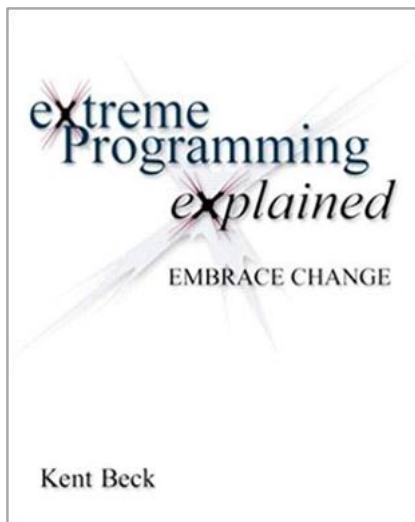
- Extreme Programming (XP)
- Scrum
- Kanban

Extreme Programming (XP)

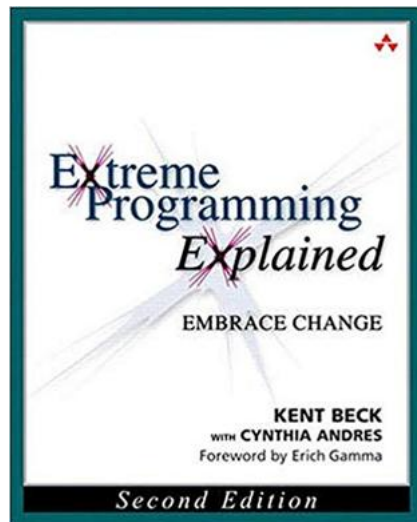
Extreme Programming



Kent Beck



1999



2004

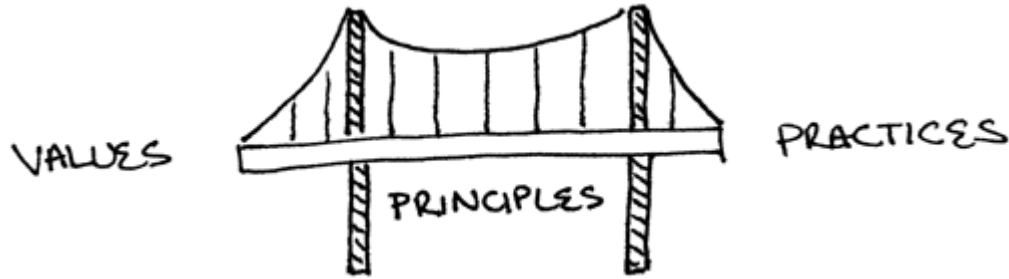
XP = Values + Principles + Practices

Values

- Communication
- Simplicity
- Feedback
- Courage
- Respect
- Quality of life (40-hour week)

Values or culture are fundamental in software!

XP = Values + Principles + Practices



Principles

- Economics
- Continuous Improvement
- Failures Happen
- Baby Steps
- Personal Responsibility

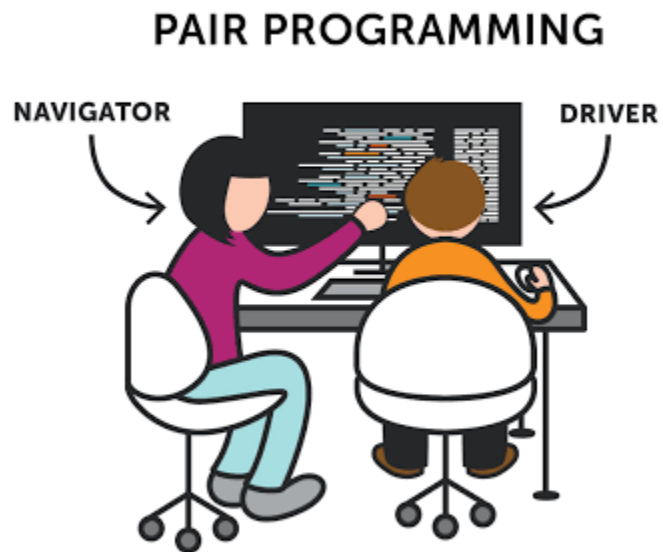
XP = Values + Principles + Practices

Process Practices	Programming Practices	Project Management Practices
Customer Representative User Stories Iterations Releases Release Planning Iteration Planning Planning Poker Slack	Incremental Design Pair Programming Automated Tests Test-Driven Development (TDD) Automated Building Continuous Integration	Working Environment Open Scope Contracts Metrics

We will study in Scrum

Tests and TDD: Chapter 8
CI: Chapter 10

Pair Programming



Survey with Microsoft Engineers (2008)

Table 1: Pair Programming benefits

1. Fewer Bugs	66
2. Spreads Code Understanding	42
3. Higher Quality Code	48
4. Can Learn from Partner	42
5. Better Design	30
6. Constant Code Reviews	22
6. Two Heads are Better than One	22
8. Creativity and Brainstorming	17
9. Better Testing and Debugging	14
10. Improved Morale	13

Survey with Microsoft Engineers (2008)

Table 2: Pair programming problems

1. Cost efficiency	79
2. Scheduling	31
3. Personality clash	25
4. Disagreements	24
5. Skill differences	22
6. Programming style differences	13
7. Hard to find a partner	12
8. Personal style differences	11
9. Distractions	10
10. Misanthropy	9
10. Bad Communication	9
10. Metrics/Hard to Reward Talent	9

Mob Programming

- The entire team collaborates on the same programming task
- Objective: group learning, knowledge dissemination, and prevent the formation of knowledge silos



Software Contracts

- Software can be developed:
 - Internally
 - Externally (outsourced) \Rightarrow requires a contract
- Types of software contracts:
 - Closed Scope
 - Negotiated Scope \Rightarrow defended by XP

Closed Scope Contracts

- Scope (features) \Rightarrow defined by the customer
- Price and deadline \Rightarrow defined by the software agency

Negotiated Scope Contracts

- Payment is per person/hour
- Scope defined in each iteration
- Contract renewed after each iteration

Negotiated Scope Contracts

- Requires maturity and customer involvement
- Advantages:
 - Fosters quality
 - Avoids delivering solely to avoid penalties
 - Clients can easily change suppliers

Exercises About XP

1. Why is XP a methodology aimed specifically at software development projects?

Scrum

Scrum

- Proposed by Jeffrey Sutherland and Ken Schwaber

SCRUM Development Process

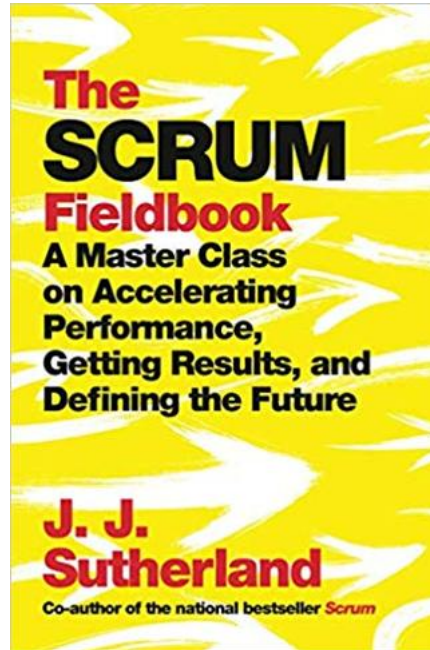
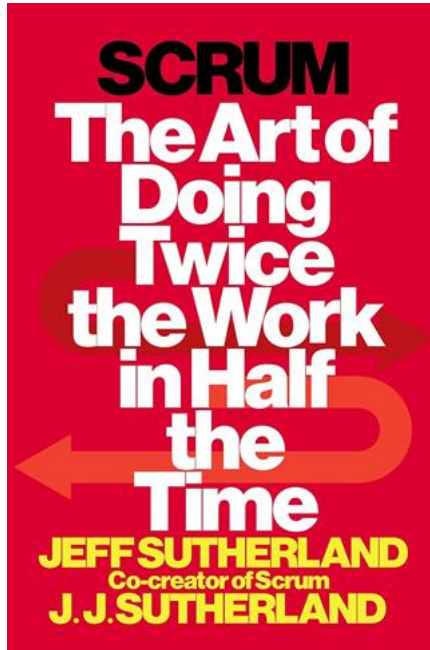
Ken Schwaber

Advanced Development Methods
131 Middlesex Turnpike Burlington, MA 01803
email virman@aol.com Fax: (617) 272-0555

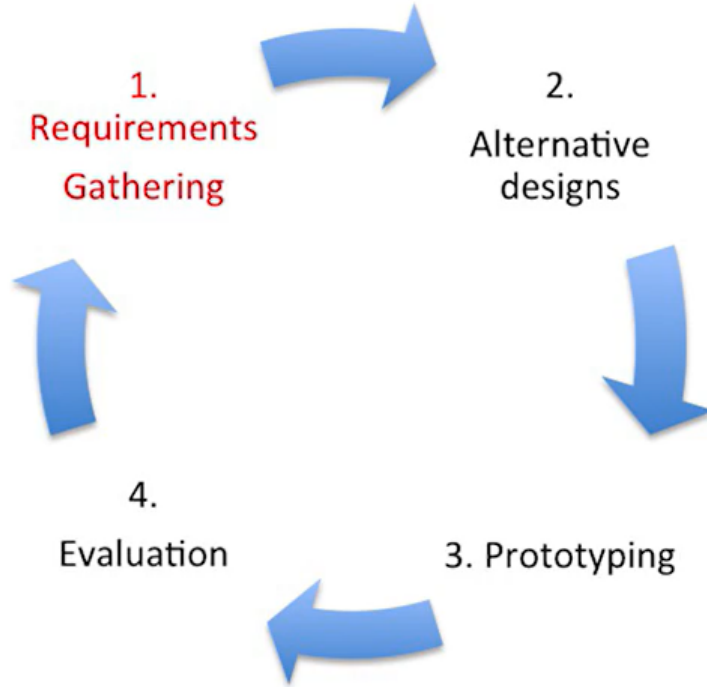
ABSTRACT. *The stated, accepted philosophy for systems development is that the development process is a well understood approach that can be planned, estimated, and successfully completed. This has proven incorrect in practice. SCRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle.*

KEY WORDS: *SCRUM SEI Capability-Maturity-Model Process Empirical*

Scrum has many books, consulting, certifications, etc

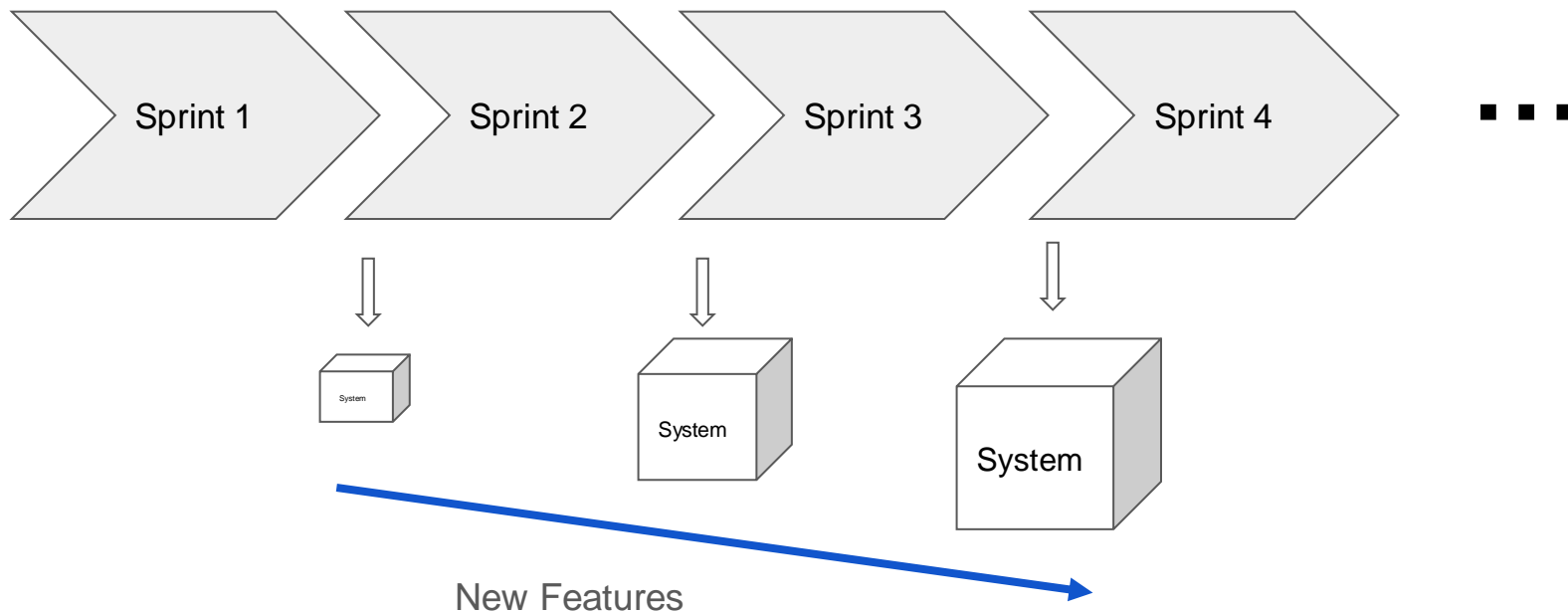


User Interface Design Cycle



Main event: Sprints

- Up to 1 month, usually 15 days



Issue vs Story vs Task

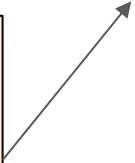
Type	Definition	When to Use?
Issue	A generic term for any work item in Jira (Story, Task, Bug, Epic, etc.).	Used for tracking work, defects, or improvements.
Story	A user requirement written from the user's perspective. Describes a feature that delivers value.	"As a user, I want to [feature], so that I can [benefit]."
Task	A work unit that contributes to completing a Story or Epic. More technical and implementation-focused.	Used for specific development, testing, or design activities.

What is done in a sprint?

- Team implements user stories
- User stories \Rightarrow features
 - example
- Example from a Q&A Forum:

A logged-in user should be able to post questions. Since it's a programming forum, questions may incorporate code blocks, which must be presented in a differentiated layout.

User stories are written on cards




Backlog



PLANNING

 Timeline

 Backlog

 Board

 Calendar

 List

 Goals

 Issues

+ Add view

DEVELOPMENT

 Code

 Project pages

Q Search



+2



Epic ▾

☐ ▾ **SCRUM Sprint 1** 6 Mar – 13 Mar (4 issues)

0 0 0

Complete sprint



focus group created

☒ **SCRUM-7** identify project user / stakeholders

TO DO ▾

-



☒ **SCRUM-6** requirements gathering

TO DO ▾

-



☒ **SCRUM-5** create focus group

TO DO ▾

-



☒ **SCRUM-8** Conduct a user survey

TO DO ▾

-



+ Create issue

☐ ▾ **SCRUM Sprint 2** 13 Mar – 20 Mar (0 issues)

0 0 0

Start sprint



Plan a sprint by dragging the sprint footer down below some issues, or by dragging issues here.

+ Create issue



ux-design-07
Software project

PLANNING



Timeline



Backlog



Board



Calendar



List



Goals



Issues



Add view

DEVELOPMENT



Code



Project pages



Add shortcut



Project settings

Projects / ux-design-07

SCRUM Sprint 1

focus group created

Q Search



🕒 4 days



Complete sprint



GROUP BY

None ▾



TO DO 4

identify project user /
stakeholders

✓ SCRUM-7

S

requirements gathering

✓ SCRUM-6

M

create focus group

✓ SCRUM-5

S

Conduct a user survey

✓ SCRUM-8

M

+ Create issue

IN PROGRESS

DONE ✓



Who writes the stories?

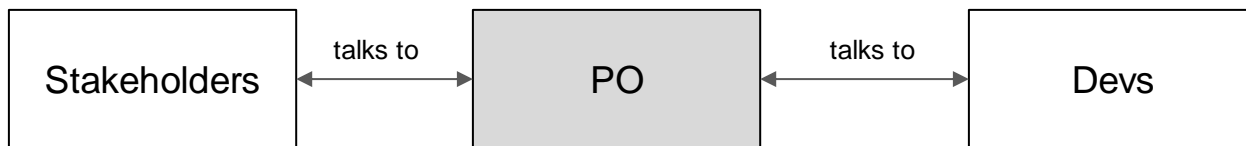
- Product Owner (PO): mandatory role in Scrum
- Expert in the problem domain

Waterfall

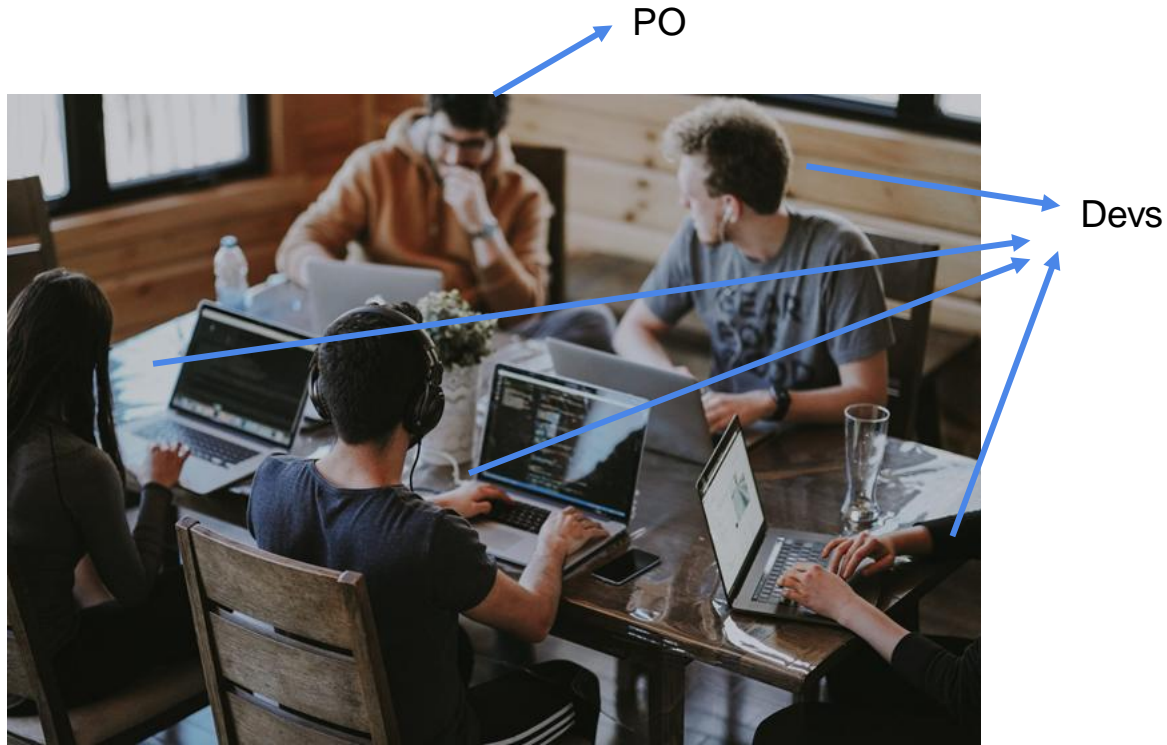


Natural language
(could take years to get ready)

Scrum



- During the sprints, PO explains stories to devs
- We change from formal/written to informal/verbal specs



Product Owner sits with developers and explains the user stories to them

What does a PO do?

- Write the user stories
- Explain the user stories to the devs
- Define the "acceptance tests" for the user stories
- Prioritize the user stories

Product Backlog

- List of user stories (and other important work items)
- Two characteristics:
 - Prioritized: top stories have higher priority
 - Dynamic: stories can come and go

Summarizing

- Iteration: sprint
- Roles: Product Owner (PO) and devs
- Artifact: product backlog

Which stories will be implemented in the next sprint?

- Decision taken at the start of the sprint
- In a meeting called **sprint planning**:
 - PO proposes stories they'd like to see implemented
 - Devs decide if they have the **velocity** to implement them

Important

- In Scrum teams, everyone is at the same hierarchical level
- The PO is not the manager of the Devs, but an expert in the product domain
- Devs, as the technical experts, can say they won't be able to implement everything the PO wants in a single sprint

Sprint Planning

- 1st part: team defines the stories of the sprint
- 2nd part: stories are broken down into tasks, which are allocated to devs

Example: Q&A Forum

Product Backlog

Story

Register user

Post questions

Post answers

Opening screen

Gamify questions/answers

Search questions/answers

Add tags

Comment on questions/answers

Product Backlog

Story

Register user

Post questions

Post answers

Opening screen

Gamify questions/answers

Search questions/answers

Add tags

Comment on questions/answers

stories selected
for the next sprint

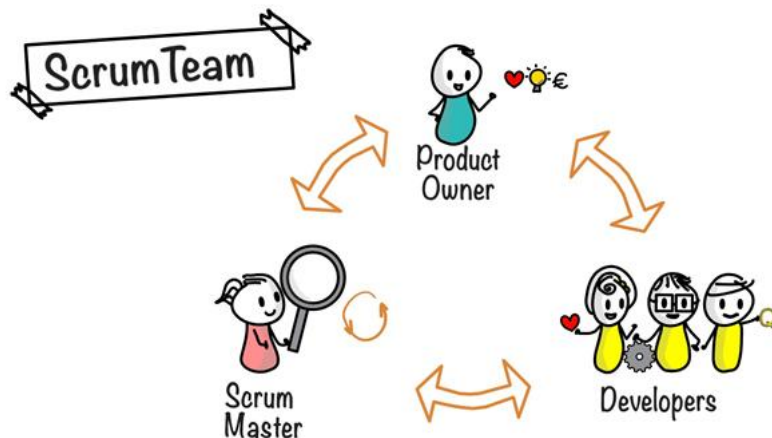
Sprint Backlog: tasks of the selected stories

- Install the database and create initial tables
- Install Node.js and Express
- Create and test a route using Express
- Implement the question page in the frontend
- Implement the backend logic for creating questions
- Implement the answer page in the frontend
- Implement the backend logic for answering question

Sprint is ready to start!

Scrum Teams

- Small (size of a basketball to a football team)
- Including 1 PO and 1 Scrum Master
- Cross-functional: devs, UX designers, data scientists, etc.



Scrum Master (SM)

- Expert who helps the team to follow Scrum
- SM is not the manager of the team, but a servant leader
- “Remover” of non-technical impediments
 - Example: developers don't have good computers
- SM can also collect process metrics
- SM can be part of more than one team

More Scrum Events

Daily Meetings (15 min)

- Each participant answers three questions:
 - What I did yesterday
 - What I intend to do today
 - What obstacles I'm facing (if any)
- Goals: Improve communication & anticipate problems



Sprint ends with two events:
Review and Retrospective

Review

- Team shows the sprint's outcome to PO and stakeholders
- Implementation of each story can be:
 - Approved
 - Partially approved
 - Rejected
- In the last two cases, it goes back to the product backlog

Retrospective

- Last event of the sprint
- Team gathers to discuss two questions:
 - What went well in the sprint?
 - How can we improve?
- Goal: continuous improvement
- It should be a blameless meeting

Exercises

1. Why is Scrum usually defined as a framework? For example, see the [Scrum Guide](#) definition:

"Scrum is a **lightweight framework** that helps people, teams and organizations generate value through adaptive solutions for complex problems."

1. Why is the Scrum framework defined as lightweight?

Exercise for Discussion

3. At what number of developers should a company start being concerned about processes like Scrum?

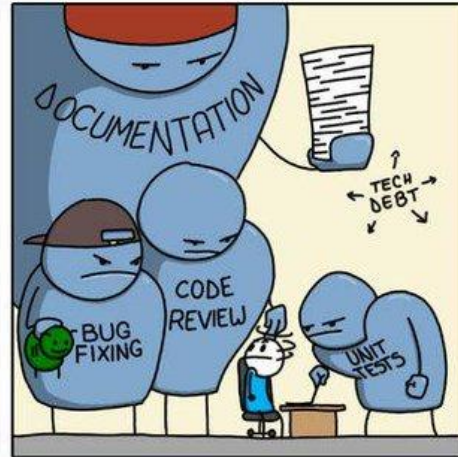
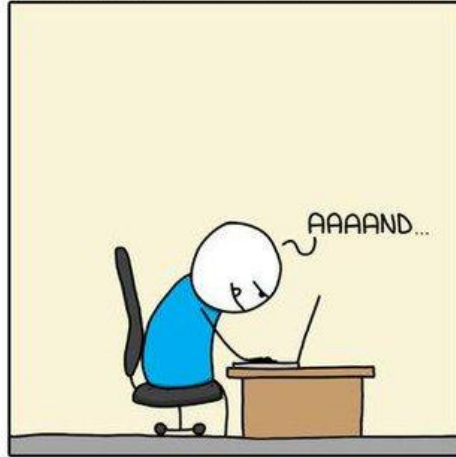
More Scrum Concepts

Time-box: all events have a well-defined duration

Event	Time-box
Sprint Planning	maximum of 8 hours
Sprint	less than 1 month
Daily Stand-up	15 minutes
Sprint Review	maximum of 4 hours
Retrospective	maximum of 3 hours

FEATURE COMPLETE

MONKEYUSER.COM



Done Criteria: used to consider stories done

- Also called DoD (Definition of Done)
- Example:
 - Unit tests with coverage $\geq 75\%$
 - Code review by another dev
 - Update documentation (if API has changed)
 - Performance test (for certain stories)

Scrum Board

Backlog	To Do	Doing	Testing	Done
<div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div></div>

Example: Mozilla project (using GitHub Projects)

Smart Scheduling

Updated 13 days ago

Filter cards

100 Backlog

📄 Bug 1632870 - Store test configuration in the task definition

Added by ahal

📄 Bug 1667401 - Always schedule "new manifests" with manifest based bugbug optimizers

Added by ahal

📄 Investigate Treeherder UI/UX changes for test filtering

Added by armenzg

🚨 Handle pushes containing multiple backouts

mozci#204 opened by marco-c

enhancement

🚨 When a task/group has inconsistent classifications, use the status of the task/group on the backout to figure it out

mozci#200 opened by marco-c

enhancement

📄 Bug 1635921 - Use |mach try fuzzy| as a means to select configurations with

1 Next up

🚨 Reinstate integration tests

mozci#390 opened by marco-c

3 In progress

📄 Build a dashboard to see schedulers results over time (both # of tests and durations)

Added by marco-c

🚨 Add a way to get durations of shadow scheduler tasks

mozci#102 opened by ahal

metrics

1 linked pull request

📄 Bug 1639164 - "mach try auto" should select the best platforms to run manifests on

assigned: marco

Added by marco-c

222 Done

🔁 Make adr dependency optional

mozci#388 opened by marco-c

📄 Bug 1671422 - Add durations to group_result actions in errorssummary formatter

assigned: ahal

Added by ahal

🔁 Cache results from data sources

mozci#368 opened by marco-c

enhancement

1 linked pull request

🔁 OOMs while analyzing some pushes

mozci#366 opened by marco-c

bug

1 linked pull request

🔁 Add support for getting groups and groups results in the Treeherder data source

mozci#312 opened by marco-c

1 linked pull request

Story Points

Story Points

- Used to estimate the size of stories (empirical process)
- Help to define what will fit in a sprint
- Use is not mandatory in Scrum

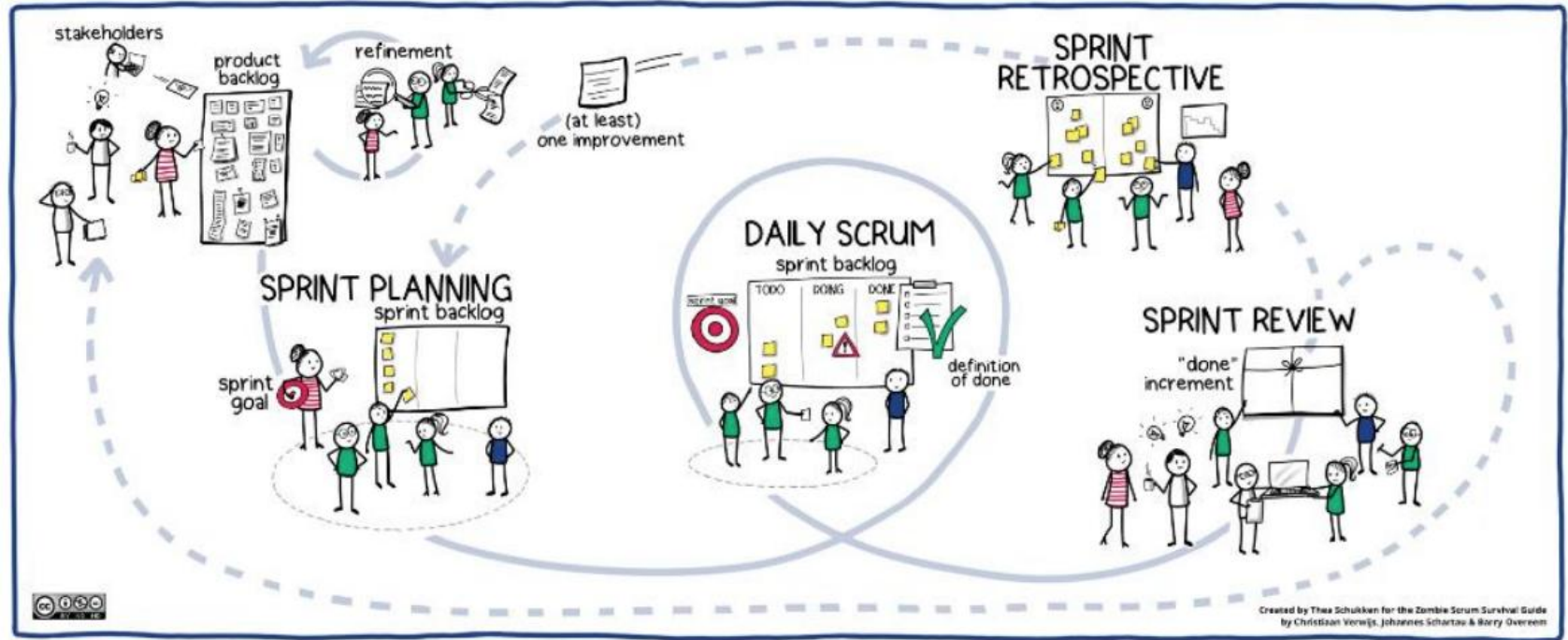
Story points scale

- Fibonacci scale: 1, 2, 3, 5, 8, 13, ...
- Team velocity: number of story points the team can implement in one sprint

Example

Story	Story Points	
Register user	8	Defined by the devs
Post questions	5	
Post answers	3	
Opening screen	5	
Gamify questions/answers	5	
Search questions/answers	8	
Add tags	5	
Comment on questions/answers	3	

Scrum in 1 slide



Interesting comment on the purpose of Scrum events

**Cory House** 
@housecor

...

You don't need a daily standup. But you do need to communicate often.

You don't need formal retrospectives. But you do need to regularly discuss improvement opportunities.

You don't need sprints. But you do need to break work down and deploy often.

You don't need a sprint review. But you do need to iterate based on feedback.

You don't need a scrum master. But you do need to assure the things above happen.

12:12 PM · Mar 11, 2023 · **195.1K** Views

181 Retweets **21** Quote Tweets **1,347** Likes

Exercises about Scrum

3. What is the difference between the top and bottom stories of the product backlog?
4. Suppose you intend to use Scrum to write a book:
 - a. What would be the items of the product backlog?
 - b. What would be the goal of the sprints?
 - c. What would be the items of the sprint backlog?
 - d. Does it make sense to have a sprint review?
 - e. Does it make sense to have a PO?

5. Suppose two teams, A and B, working on different projects, hired by different companies:

- Both teams adopt 15-day sprints
- Both teams have 5 devs
- Team A's velocity is 24 story points
- Team B's velocity is 16 story points

Can we say that Team A is 50% more productive than B?
Justify.

6. Suppose a text editor with two stories:

- H1: As a user, I want to open a given file in the editor
- H2: As a user, I want to edit the currently opened file

The PO has firmly prioritized H2 for one sprint and H1 for a subsequent sprint.

(a) As a dev, would you follow the PO's priority?

(b) If yes, how would you implement the editing of a file (H2) without first implementing its opening (H1)?

Kanban

Kanban

- Originated in the 1950s in Japan
- Toyota Production System
- Lean manufacturing, just-in-time production, etc

Kanban = "visual card"



Kanban in Software Development



Kanban vs Scrum

- Kanban is simpler
- No sprints
- It's not mandatory to have roles and events, including:
 - Scrum master
 - Daily Scrum, Retrospectives, Reviews
- Team defines roles and events

Kanban Board

Large columns in the board: kanban steps

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	in progress	ready	in progress	ready	in progress	done

1st sub-column

2nd sub-column

We will explain shortly


Time →

Kanban is a Pull System

- Members:
 - a. Pull a task to work on
 - b. Complete the task and move it forward on the board
 - c. Go back to step (a)




time

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	in progress	ready	in progress	ready	in progress	done




time


Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	em espec.	ready	in progress	ready	in progress	done





Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	in progress	ready	in progress	ready	in progress	done

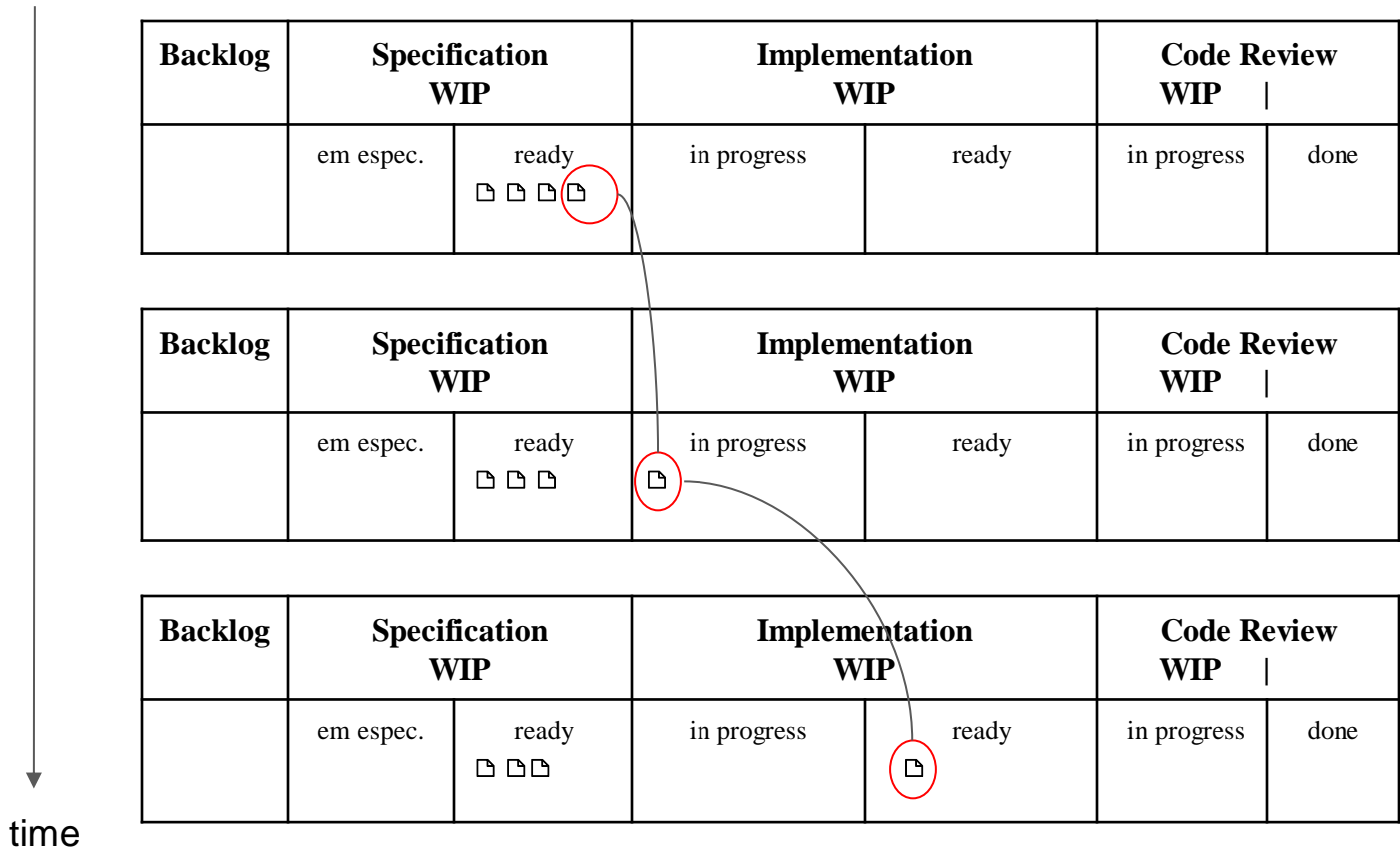


time

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	em espec.	ready	in progress	ready	in progress	done

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	em espec. 	ready	in progress	ready	in progress	done

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
	em espec.	ready    	in progress	ready	in progress	done



Example 2

Yesterday

Backlog	Specification WIP		Implementation		Code Review WIP	
H3	in progress H2	ready T6 T7 T8 T9	in progress T4 T5	ready T3	in progress T2	done T1

Today

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
H3	in progress	ready T8 T9 <u>T10</u> <u>T11</u> <u>T12</u>	in progress T4 T5 <u>T6</u> <u>T7</u>	ready	in progress <u>T3</u>	done T1 <u>T2</u>

WIP Limits

Work in Progress (WIP) Limits

Backlog	Specification 2		Implementation 5		Code Review 3	
H3	in progress	ready T8 T9 T10 T11 T12	in progress T4 T5 T6 T7	ready	in progress T3	ready T1 T2

WIP Limits

- Maximum number of tasks per step
- Counting: in progress + ready

Backlog	Specification 2		Implementation 5		Code Review 3	
H3	in progress	ready T8 T9 T10 T11 T12	in progress T4 T5 T6 T7	ready	in progress T3	ready T1 T2

4 + 0

Goals of WIP Limits

- Keep a sustainable workflow
- Prevent the team from becoming overloaded with work
 - WIP: agreement between the team and organization on the work capacity of a team
- Prevent work from being concentrated in one step

Common recommendation:

Stop starting, start finishing

Implementation is at the limit, thus it's time to review

Backlog	Specification (2)		Implementation (5)		Code Review (3)	
			X X	X X X		

WIP of Specification Step

- Stories in progress + number of groups of tasks (rows on the 2nd subcolumn)
- Tasks in the same row = resulting from the same story

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
H3	in progress	ready T8 T9 T10 T11 T12	in progress T4 T5 T6 T7	ready	in progress T3	done T1 T2

0 +1 +1

WIP of Code Review: only tasks in progress

Backlog	Specification WIP		Implementation WIP		Code Review WIP	
H3	in progress	ready T8 T9 T10 T11 T12	in progress T4 T5 T6 T7	ready	in progress T3	done T1 T2

+1

Does not count for
WIP purposes

Final Comments on Kanban

- Simpler than Scrum
 - Recommended for mature teams
 - Perhaps, start with Scrum and then move to Kanban
- Evolutionary method:
 - Start with the current flow
 - Understand the problems and bottlenecks
 - Propose small and gradual improvements

Exercises on Kanban

1. What is the error in the following Kanban board?

Backlog	Specification (2)		Implementation (5)		Review (3)	
			X	X		
			X	X		
			X	X		

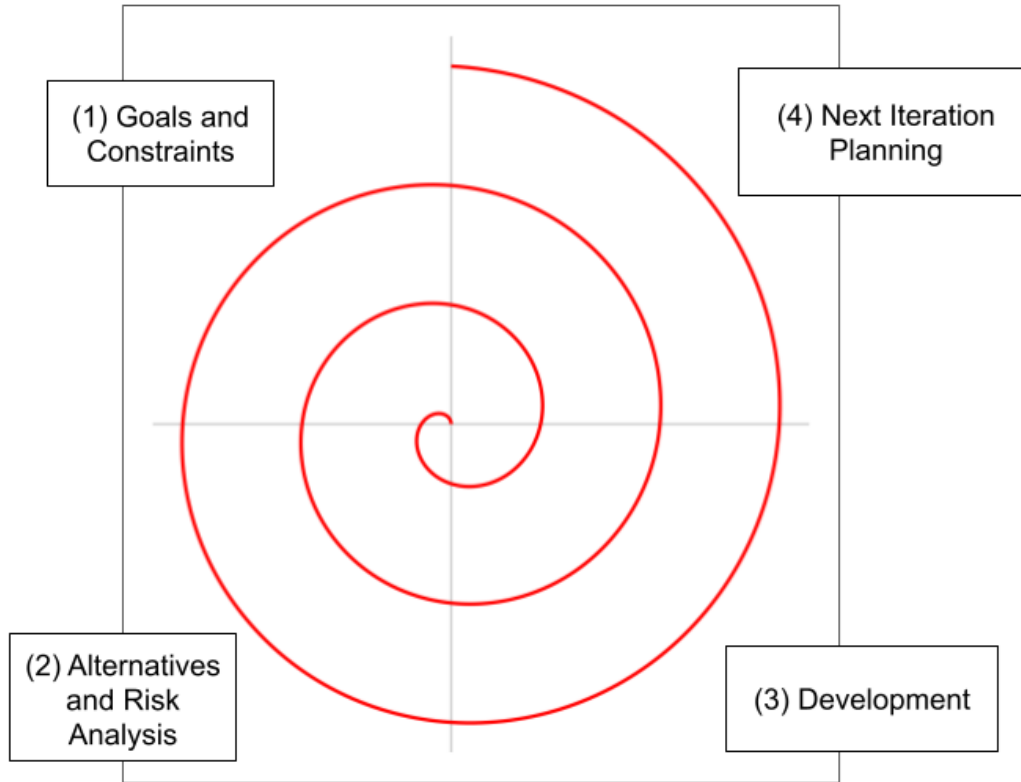
2. Is it possible to move a card back on a Kanban board? If so, describe a situation in which this could occur.
3. Work overload is a common problem in software teams. How can Kanban help solve this problem?
4. Another problem in software teams is developers who rush to deliver stories, but without the proper quality level. How can Kanban help solve this problem?

Non-Agile Processes

Iterative Methods

- Transition Waterfall (~1970) to Agile (~2000) was gradual
- Iterative or evolutionary methods were proposed, before the dissemination of agile principles
- Examples:
 - Spiral Method (1986)
 - Rational Unified Process (2003)

Spiral Model

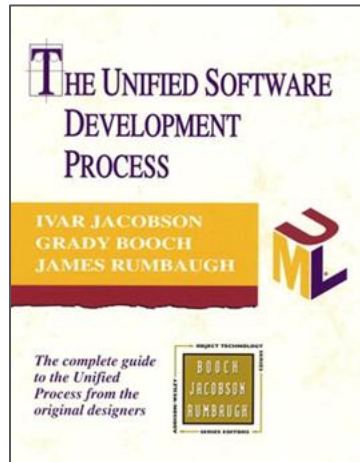


Proposed by Barry Boehm

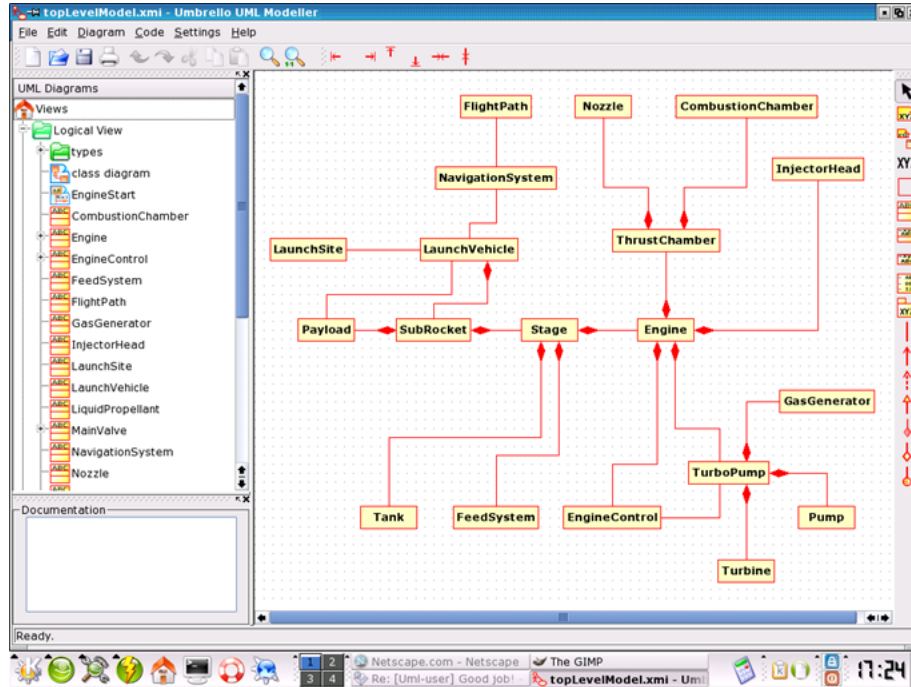
Iterations: 6 to 24 months (then, longer than in XP or Scrum)

Rational Unified Process (RUP)

- Rational was a company acquired by IBM
- Key characteristic: plan-and-document, using UML and CASE tools



CASE: Computer-Aided Software Engineering



Name comes from CAD systems
(used in traditional engineering)

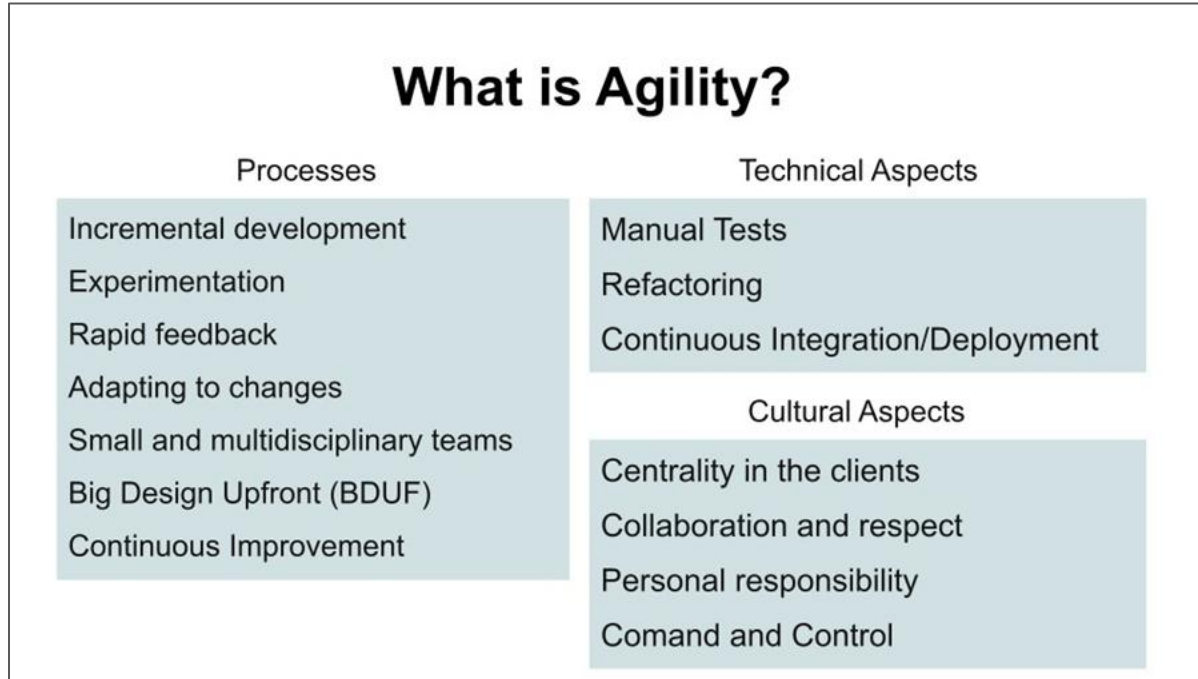


Before concluding

- Processes are not used 100% as in the textbooks
- Experimentation is important!

Exercises

1. This slide groups agile principles in three areas: processes, technical aspects, and cultural aspects. However, in each area there is a characteristic **not** compatible with agility. Indicate such characteristic.



2. Suppose that your university plans to migrate to a new learning management system. It is considering three strategies:

(a) Develop the new system internally, using devs from the university.

(b) Outsource the development to a software agency.

(c) Buy or subscribe to an existing product on the market.

Assuming that the system in the three options will be developed using Scrum, describe the most suitable Product Owner profile for each option.

3. In Scrum, user stories are not the only possible items in a product backlog. For instance, consider the following types of bugs:

- (a) A bug detected during the implementation of a sprint's user story.
- (b) A non-critical but very complex bug reported by a user.
- (c) A critical bug that is affecting several users.

Which of these bugs should be added to the product backlog?

4. This question is similar to the previous one but focuses on refactorings. Thus, consider the following types of refactorings:

- (a) A refactoring that can be performed in a few minutes.
- (b) A complex refactoring that changes the system's architecture.

Which of these refactorings should be added to the product backlog?

5. There are four important variables in software contracts: scope, time, cost, and quality.

XP argues that it is impossible to fix these four variables via a contract, as surprises will occur during the project.

Thus, suppose a fixed scope contract. If a surprise occurs during the project, which of these variables is likely to be sacrificed by the contracting company to avoid penalties?

6. Suppose you are the tech lead of a team.

The developers are complaining that they are unable to use certain modules because the documentation of their public interfaces is outdated.

Upon investigating the issue, you confirmed that developers often change the modules' interfaces, but do not update the documentation.

Assuming the team uses Scrum, what measure would you take to prevent this problem?

7. In SE, anti-patterns are solutions not recommended for a certain problem. Thus, describe three anti-patterns for a Product Owner.

End