

## 0.1 Scrum

### SCRUM FRAMEWORK

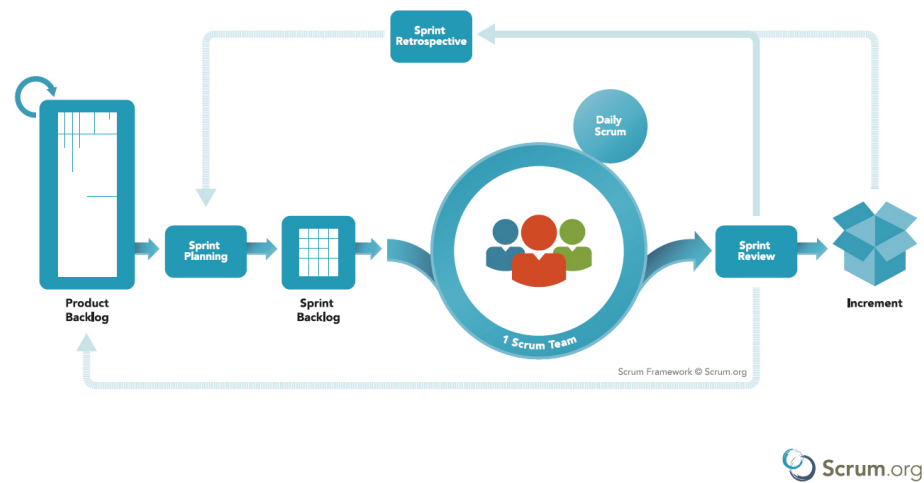


Figure 1: Scrum Overview

#### 0.1.1 Events

- **Sprint planning meeting**  
Review the features for the next Sprint
- **Daily scrum**  
Daily stand-up meeting for coordination and commitment among peers
- **Sprint review**  
The team presents what it accomplished during the sprint
- **Sprint retrospective**  
Team discusses what they'd like to start/stop/continue doing

#### 0.1.2 Artifacts

- **Product backlog**  
A list of all desired work on the project
- **Sprint backlog**  
Shows list of tasks and estimates of work remaining (hours)
- **Sprint burndown chart**  
Shows, during a sprint, the total work remaining per day

### 0.1.3 Roles

- Product Owner
  - Define the features of the product and priorities
  - Decide on release date and content
  - Accept or reject work results
- Scrum Master
  - Enact Scrum values and Practices
  - Remove impediments and external interferences
  - Ensure that the team is fully functional and productive
- Development Team
  - Does the work
  - Self-organizing
  - Typically 5-9 people, ideally full time and multifunctional

### 0.1.4 Agile Estimation

**User story** Describes something of value to the user or the system

Example

**As a** student, **I want to** indicate preferences for colleagues to share the same scholar timetable, **so that** I can be more productive in group works.

**Story points** Relative measure for expressing the “size” of a user story, Influenced by difficulty, risk, complexity, etc. Typically exponential.

**Team velocity** The number of story points implemented per Sprint

## 0.2 eXtreme Programming (XP)

Developed by Kent Beck.

### 0.2.1 Core Values

- Communication
- Simplicity
- Feedback
- Courage

### 0.2.2 Practices

- The Planning Game
  1. The customer comes up with a list of desired features, that are aggregated as user stories (similarly to Scrum).
  2. The developers sort them using story points, so as to know which are easier/harder to implement.
  3. Using this information and project velocity (total story points done per iteration), the customer prioritizes which features to implement.
- Small Releases
  - Start with the smallest useful feature set
  - Release early and often, adding a few features each time
  - Releases can be date driven or user story driven
- System Metaphor

The system metaphor is a story that everyone - customers, programmers, and managers - can tell about how the system works.
- Simple Design

Use the simplest possible design that gets the job done, so that there are obviously no deficiencies
- Test-driven Development

Write tests *before* adding a feature, or before fixing a bug. Use unit and acceptance tests.
- Refactoring

Improve the structure of the code without changing externally visible behavior (e.g removing duplicate code)

Refactoring is heavily related to automated tests and simple design.
- Pair Programming

Process:

  - Two programmers work together at one machine
  - Driver enters code, while navigator critiques it
  - Periodically switch roles and pairs
  - Requires proximity in lab or work environment

Advantages:

  - Serves as an informal review process

- Helps developing collective ownership and spread knowledge
  - Improves quality, whilst maintaining (or improving) productivity
- Collectice Code Ownership  
Any developer can work on any part of the code base at any time
- Continuous Integration  
All changes are integrated into the code base at least daily
- Sustainable Pace  
“Fresh and eager every morning, and tired and satisfied every night”
- On-Site Customer  
Development team has continuous access to a real live customer, that is, someone who will actually be using the system, or a proxy (in Scrum: product owner)
- Coding Standards  
Everyone codes to the same standards