Software Engineering

Part 4 – Software Engineering Process

ICM – Computer Science Major – Software Engineering - Part 1: Introduction
M1 Cyber Physical and Social Systems – CPS2 engineering and development - Part 3: Software Engineering
Guillaume Muller
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/softeng/

Software engineering methods – definition

software engineering method
organized and systematic approach to developing software for a target computer

— ISO/IEC TR 19759:2015 Software Engineering Body of Knowledge (SWEBOK)

objectives:

- facilitate human understanding, communication, and coordination
- aid management of software projects
- measure and improve the quality of software products in an efficient manner
- support process improvement
- provide a basis for automated support of process execution

Software *development* life cycles (SDLC)

software development life cycle process
software processes used to specify and transform software requirements into a deliverable software product

— ISO/IEC TR 19759:2015 Software Engineering Body of Knowledge (SWEBOK)

history

1970s
- Structured programming since 1969
- Cap Gemini SDM, originally from PANDATA, the first English translation was published in 1974. SDM stands for System Development Methodology

1980s
- Structured systems analysis and design method (SSADM) from 1980 onwards
- Information Requirement Analysis/Soft systems methodology

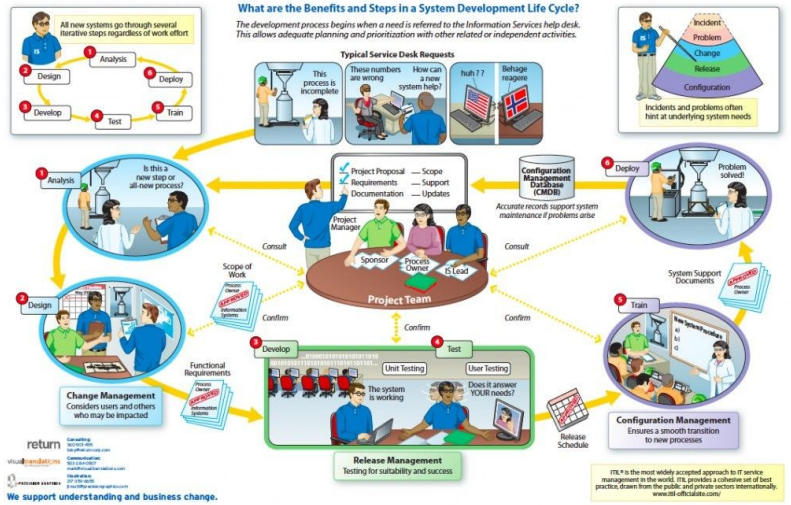
1990s
- Object-oriented programming (OOP) developed in the early 1960s, and became a dominant programming approach during the mid-1990s
- Rapid application development (RAD), since 1991
- Dynamic systems development method (DSDM), since 1994
- Scrum, since 1995
- Team software process, since 1998
- Rational Unified Process (RUP), maintained by IBM since 1998
- Extreme programming, since 1999

2000s
- Agile Unified Process (AUP) maintained since 2005 by Scott Ambler
- Disciplined agile delivery (DAD) Supersedes AUP

2010s
- Scaled Agile Framework (SAFe)
- Large-Scale Scrum (LeSS)
- DevOps

source https://en.wikipedia.org/wiki/Software_development_process

Example of a software development life cycle



Analysis vs. Design
What vs. How

During Analysis

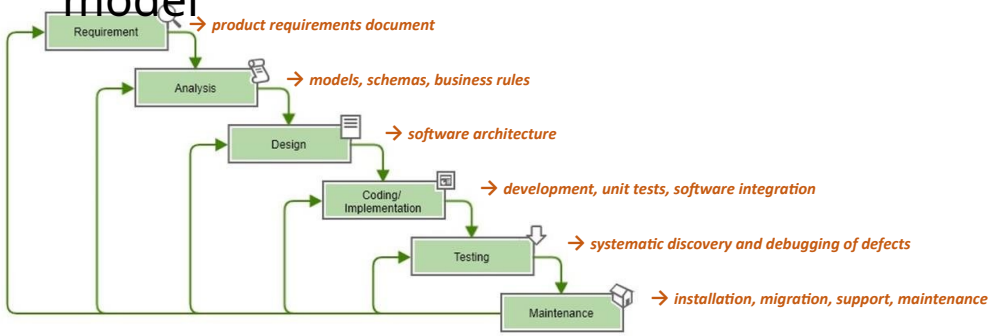
- To know about the application domain and the requirements
- Development of a coarse-grained model to show where responsibilities are, and how objects interact
- Models show a message being passed, but no worry too much about the contents of each message

During Design

- To know how the software should work
- Development of fine-grained models to show exactly what will happen when the system runs

W.W. Royce, 1970

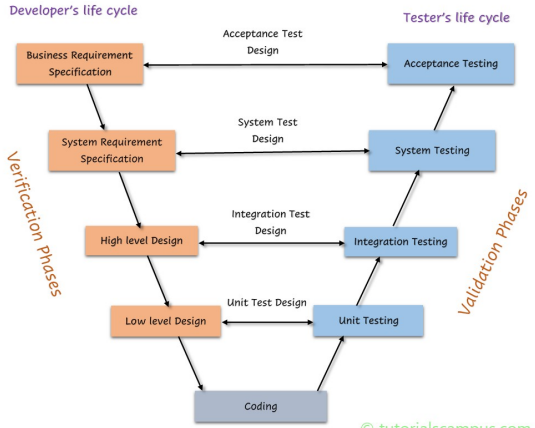
The waterfall development life cycle model



Many variants of this model

☺ Well-documented and well structured
☺ Easy to maintain
☹ No prototype, late feedback to customer
☹ Major project risks, e.g. Implementation technology, are faced at the end of the project

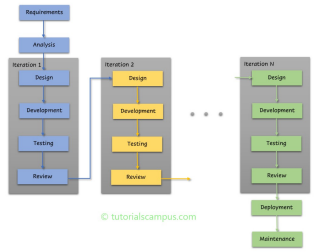
The V-model life cycle model



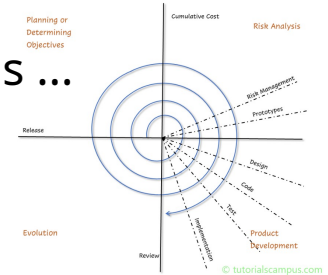
source: https://www.tutorialscampus.com/sdlc/v-model.htm

1980s

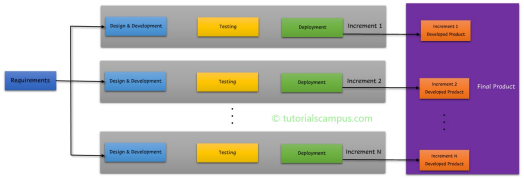
Many more SDLC models …



iterative SDLC model



spiral SDLC model

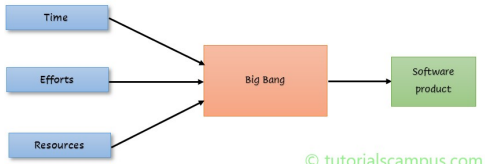


incremental SDLC model

source: https://www.tutorialscampus.com/sdlc/

1990s

Simplest SDLC models …



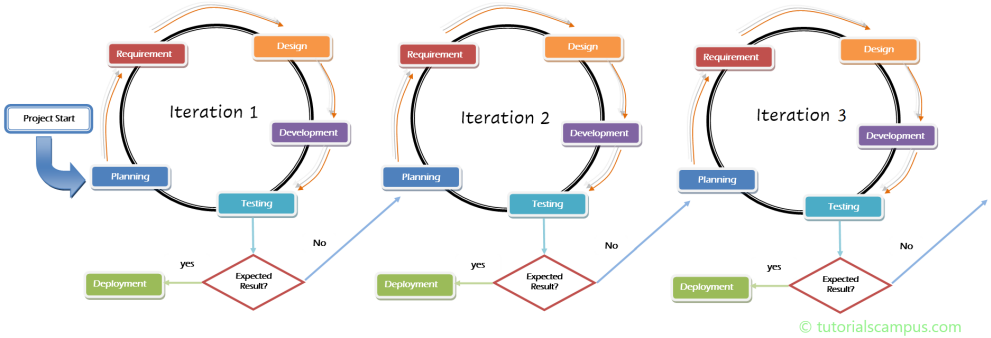
Big Bang methodology

One main rule:
always resolve the most important issue first

Chaos model

source: https://www.tutorialscampus.com/sdlc/

Agile methods **(not a SDLC model!)**



Many Agile SDLC models

Lightweight methods; short, iterative development cycles; self-organizing teams; simpler designs; code refactoring; test-driven development; frequent customer involvement; create demonstrable working product with each development cycle

source: https://www.tutorialscampus.com/sdlc/agile-model.htm

2001

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.

4 values

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

source: https://agilemanifesto.org/

2001

Principles behind the Agile Manifesto

We follow these 12 principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

4 values and 12 principles

source: https://agilemanifesto.org/

Agile vs « traditional » methods

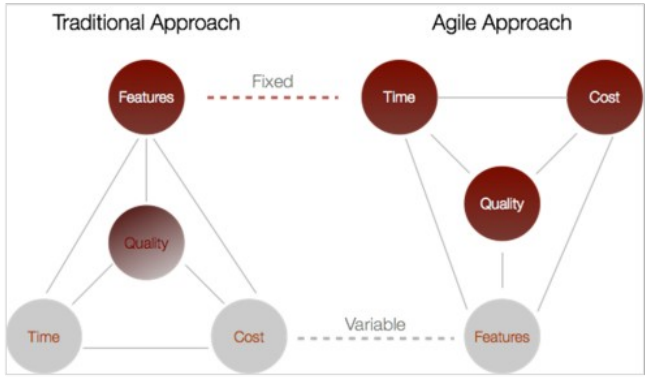


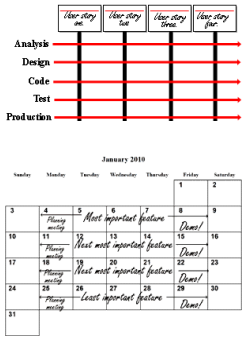
Figure 1. Comparison of the project triangles for traditional and Agile approaches (*source:* Awad, 2012; Beck et al., 2001)

Agile development methods

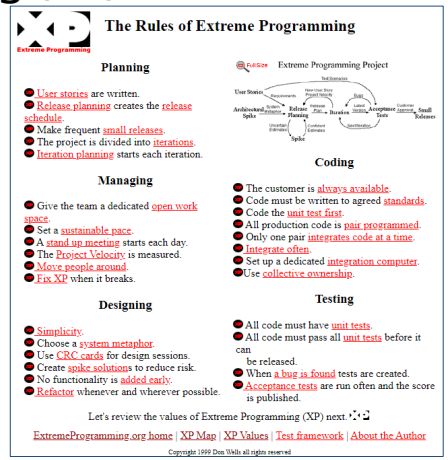
- Dynamic System Development Methodology and RAD (www.dsdm.org, 1995)
- Scrum (Sutherland and Schwaber, 1995)
- XP - eXtreme Programming (Beck, 1999)
- Feature Driven Development (DeLuca, 1999)
- Adaptive Sw Development (Highsmith, 2000)
- Lean Development (Poppendieck, 2003)
- Crystal Clear (Cockburn, 2004)
- Agile Unified Process (Ambler, 2005)
- DevOps (

Extreme Programming (XP)

Manage Goals Instead of Activities



The values of XP
simplicity
communication
feedback
respect
courage



The Rules of Extreme Programming

Planning
- User stories are written.
- Release planning creates the release schedule.
- Make frequent small releases.
- The project is divided into iterations.
- Iteration planning starts each iteration.

Managing
- Give the team a dedicated open work space.
- Set a sustainable pace.
- A stand up meeting starts each day.
- The Project Velocity is measured.
- Move people around.
- Fix XP when it breaks.

Designing
- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

Extreme Programming Project

Coding
- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All production code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Set up a dedicated integration computer.
- Use collective ownership.

Testing
- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

Let's review the values of Extreme Programming (XP) next.

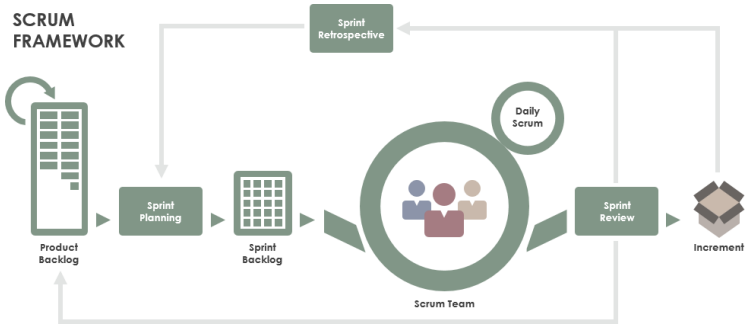
ExtremeProgramming.org home | XP Map | XP Values | Test framework | About the Author

Copyright 1999 Don Wells all rights reserved

source: http://www.extremeprogramming.org

Scrum - Framework

A **Scrum Master** fosters an environment where
1. A **Product Owner** orders the work for a complex problem into a **Product Backlog**.
2. The **Scrum Team** turns a selection of the work into an **Increment** of value during a **Sprint**.
3. The Scrum Team and its stakeholders **inspect the results and adjust** for the next Sprint.
4. *Repeat*

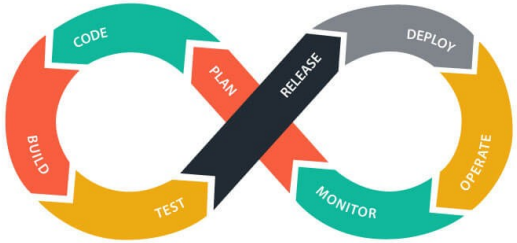


https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

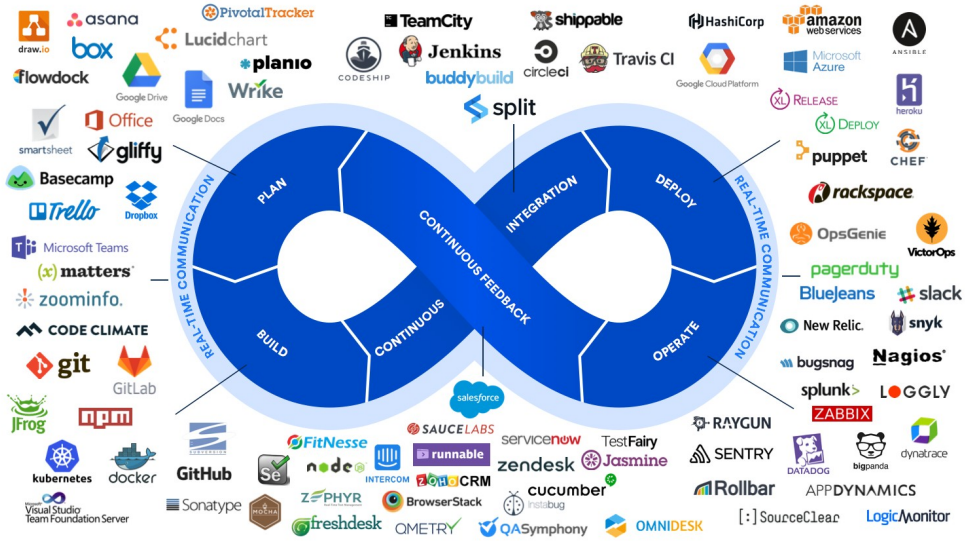
DevOps

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology.

— Contributors, Wikipedia https://en.wikipedia.org/wiki/DevOps

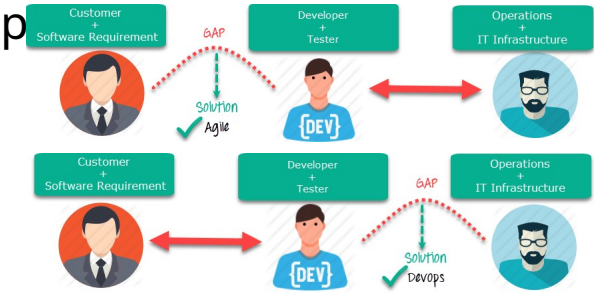


⚠ There are many variants



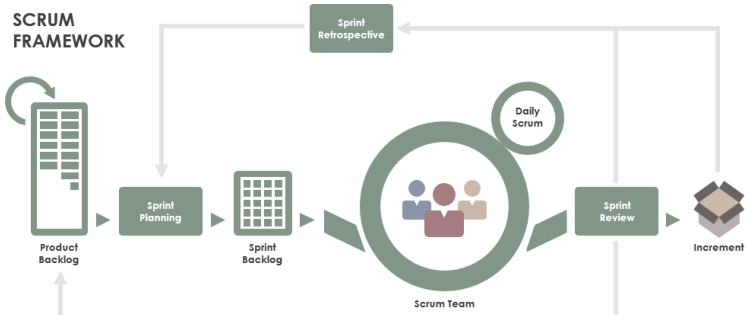
https://s32860.pcdn.co/wp-content/uploads/2019/11/devops-hero-1-87966cfbc9c5713ae047551c7b22985c.png

Agile vs DevOp



- DevOps is a practice of bringing development and operations teams together whereas Agile is an iterative approach that focuses on collaboration, customer feedback and small rapid releases.
- DevOps focuses on constant testing and delivery while the Agile process focuses on constant changes.
- DevOps requires relatively a large team while Agile requires a small team.
- DevOps leverages both shifts left and right principles, on the other hand, Agile leverage shift-left principle.
- The target area of Agile is Software development whereas the Target area of DevOps is to give end-to-end business solutions and fast delivery.
- DevOps focuses more on operational and business readiness whereas Agile focuses on functional and non-function readiness.

Source: https://www.guru99.com/agile-vs-devops.html



Software Engineering

Part 5 – Focus on the Scrum methodology

ICM – Computer Science Major – Software Engineering - Part 1: Introduction
M1 Cyber Physical and Social Systems – CPS2 engineering and development - Part 3: Software Engineering
Guillaume Muller
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/softeng/

Scrum - roles

The Scrum Team consists of one **Scrum Master**, one **Product Owner**, and **Developers**

Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint.

- Create a plan for the Sprint, the Sprint Backlog;
- Instill quality by adhering to a Definition of Done;
- Adapt their plan each day toward the Sprint Goal; and,
- Hold each other accountable as professionals.

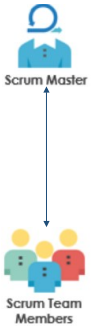
Scrum - roles

The Scrum Team consists of one **Scrum Master**, one **Product Owner**, and **Developers**

Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. How this is done may vary widely across organizations, Scrum Teams, and individuals.

- Develop and explicitly communicate the Product Goal;
- Create and clearly communicate Product Backlog items;
- Order Product Backlog items; and,
- Ensure that the Product Backlog is transparent, visible and understood.

Scrum - roles

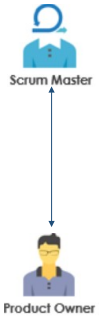


The Scrum Team consists of one **Scrum Master**, one **Product Owner**, and **Developers**

Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization.
The Scrum Master is accountable for the Scrum Team's effectiveness. They do this by enabling the Scrum Team to improve its practices, within the Scrum framework.
Scrum Masters are true leaders who serve the Scrum Team and the larger organization.
Serves the Scrum Team:

- Coaching the team members in self-management and cross-functionality;
- Helping the Scrum Team focus on creating high-value Increments that meet the Definition of Done;
- Causing the removal of impediments to the Scrum Team's progress; and,
- Ensuring that all Scrum events take place and are positive, productive, and kept within the timebox.

Scrum - roles

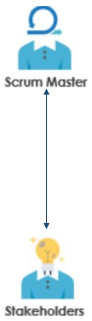


The Scrum Team consists of one **Scrum Master**, one **Product Owner**, and **Developers**

Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization.
The Scrum Master is accountable for the Scrum Team's effectiveness. They do this by enabling the Scrum Team to improve its practices, within the Scrum framework.
Scrum Masters are true leaders who serve the Scrum Team and the larger organization.
Serves the Product Owner:

- Helping find techniques for effective Product Goal definition and Product Backlog management;
- Helping the Scrum Team understand the need for clear and concise Product Backlog items;
- Helping establish empirical product planning for a complex environment; and,
- Facilitating stakeholder collaboration as requested or needed.

Scrum - roles



The Scrum Team consists of one **Scrum Master**, one **Product Owner**, and **Developers**

Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization.
The Scrum Master is accountable for the Scrum Team's effectiveness. They do this by enabling the Scrum Team to improve its practices, within the Scrum framework.
Scrum Masters are true leaders who serve the Scrum Team and the larger organization.
Serves the Organization:

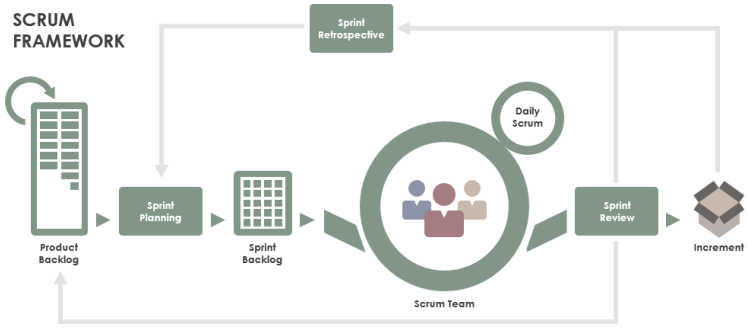
- Leading, training, and coaching the organization in its Scrum adoption;
- Planning and advising Scrum implementations within the organization;
- Helping employees and stakeholders understand and enact an empirical approach for complex work; and,
- Removing barriers between stakeholders and Scrum Teams

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum - Framework

A **Scrum Master** fosters an environment where

1. A **Product Owner** orders the work for a complex problem into a **Product Backlog**.
2. The **Scrum Team** turns a selection of the work into an **Increment** of value during a **Sprint**.
3. The Scrum Team and its stakeholders **inspect the results and adjust** for the next Sprint.
4. *Repeat*



https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum – product backlog

The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team.

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Example Product Backlog
Source: https://www.scrum-institute.org/The_Scrum_Product_Backlog.php

Scrum – sprint planning

Sprint Planning initiates the Sprint by laying out the work to be performed for the Sprint. This resulting plan is created by the collaborative work of the entire Scrum Team.

Topic One: Why is this Sprint valuable?
Product Owner proposes how to increase the value and utility of the product
Scrum Team collaborates to define **Sprint Goal**

Topic Two: What can be Done this Sprint?
Developers discuss with Product Owner and select items from the Product Backlog to include in the current Sprint
Refine items, estimate how much can be done in the Sprint timebox

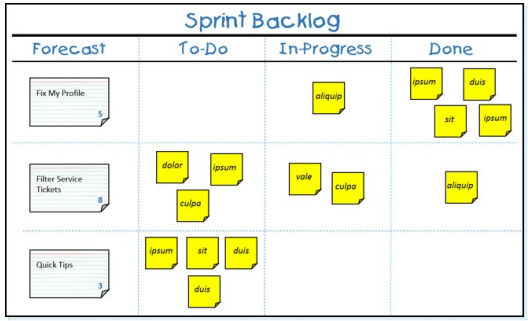
Topic Three: How will the chosen work get done?
Developers **plan** the work. Decompose Product Backlog into smaller work items

Output: **Sprint Backlog**: Sprint Goal, Product Backlog selected for the Sprint, Plan for delivering them

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum – sprint backlog

The Sprint Backlog is composed of the Sprint Goal (why), the set of Product Backlog items selected for the Sprint (what), as well as an actionable plan for delivering the Increment (how)

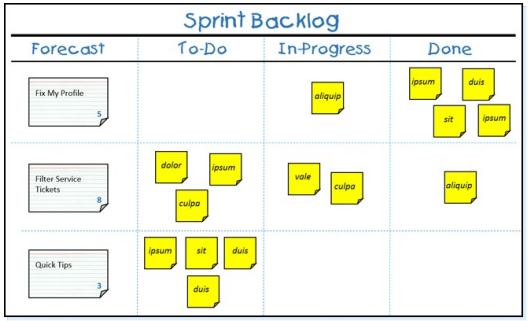


Example Sprint Backlog
Source: https://www.scrum.org/resources/what-is-a-sprint-backlog

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum – daily scrum

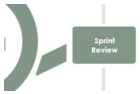
For developers only - The purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work.



Example Sprint Backlog
Source: https://www.scrum.org/resources/what-is-a-sprint-backlog

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum – sprint review



inspect the outcome of the Sprint and determine future adaptations. The Scrum Team presents the results of their work to key stakeholders and progress toward the Product Goal is discussed.

During the event, the Scrum Team and stakeholders review what was accomplished in the Sprint and what has changed in their environment. Based on this information, attendees collaborate on what to do next. The Product Backlog may also be adjusted to meet new opportunities. The Sprint Review is a working session and the Scrum Team should avoid limiting it to a presentation.

https://startinfinity.com/product-management-framework/scrum-sprint/sprint-review-vs-sprint-retrospective

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum – Increment and Done

An Increment is a concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable.

The Definition of Done is a formal description of the state of the Increment when it meets the quality measures required for the product

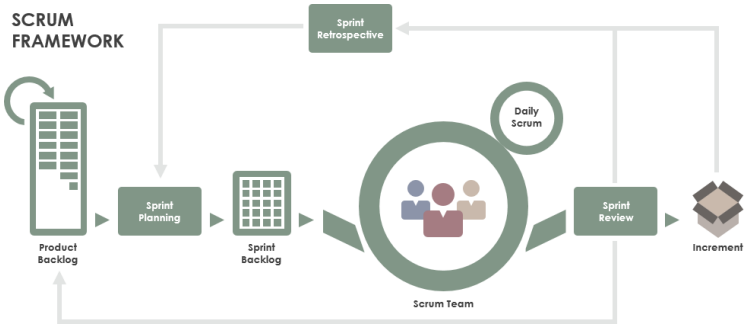
Scrum Foundations: Increment and Done (15)
source: https://youtu.be/0MTWysPmaoU

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum – Sprint retrospective

Sprint Retrospective

The purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness.
The Scrum Team inspects how the last Sprint went with regards to individuals, interactions, processes, tools, and their Definition of Done.
The Scrum Team identifies the most helpful changes to improve its effectiveness.



https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html

Scrum - values

https://www.scrum.org/ (with videos, guide)
https://www.scrumguides.org/index.html