Chapter 3

Agile Software Development

Rapid Software Development

Rapid development and delivery have become crucial requirements for software systems due to:

- **Fast-changing business environments:** It is challenging to produce stable software requirements in such environments.
- **Evolving business needs:** Software needs to adapt quickly to reflect changing business priorities.

Plan-Driven Development

Plan-driven development, while suitable for certain systems, falls short in meeting these business needs.

Agile Development Methods

In the late 1990s, agile development methods emerged with the goal of significantly reducing the delivery time for working software systems. These methods prioritize:

- Iterative development
- Customer involvement
- Adaptability to change
- Focus on delivering value quickly

Agile Methods

Agile methods emerged due to dissatisfaction with the overheads of software design methods in the 1980s and 1990s. These methods emphasize:

- Focus on code: Prioritizing the development of working code over extensive design documentation.
- Iterative approach: Breaking down development into smaller, iterative cycles.

- Rapid delivery: Aiming to deliver working software quickly and evolving it incrementally to meet changing requirements.

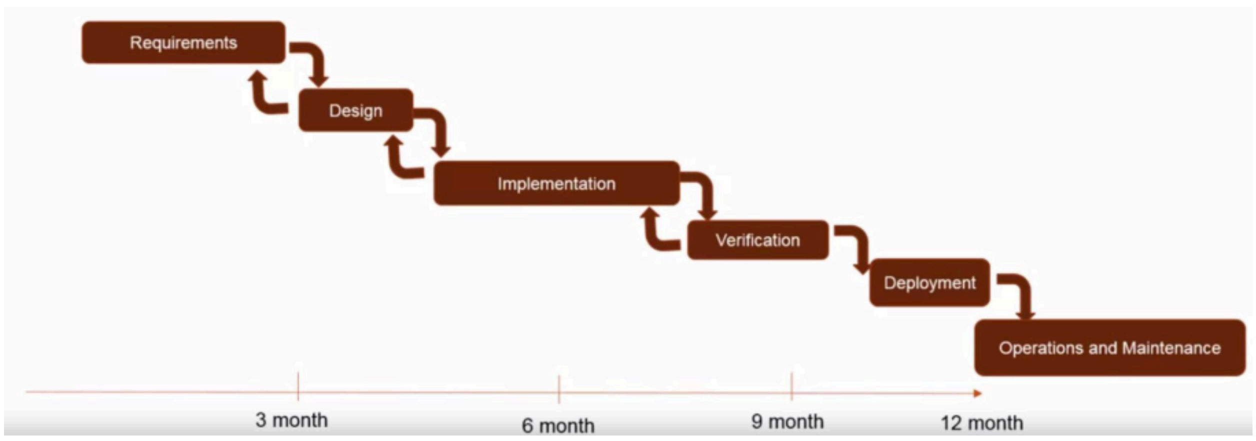
Goals

The goal of agile methods is to:

- Reduce overheads in the software process (e.g., by limiting documentation).
- Enable rapid response to changing requirements without excessive rework.

Why agile ?

- We can predict the requirements accurately
- Translation is going to be perfect



Agile Manifesto

The Agile Manifesto outlines four core values:

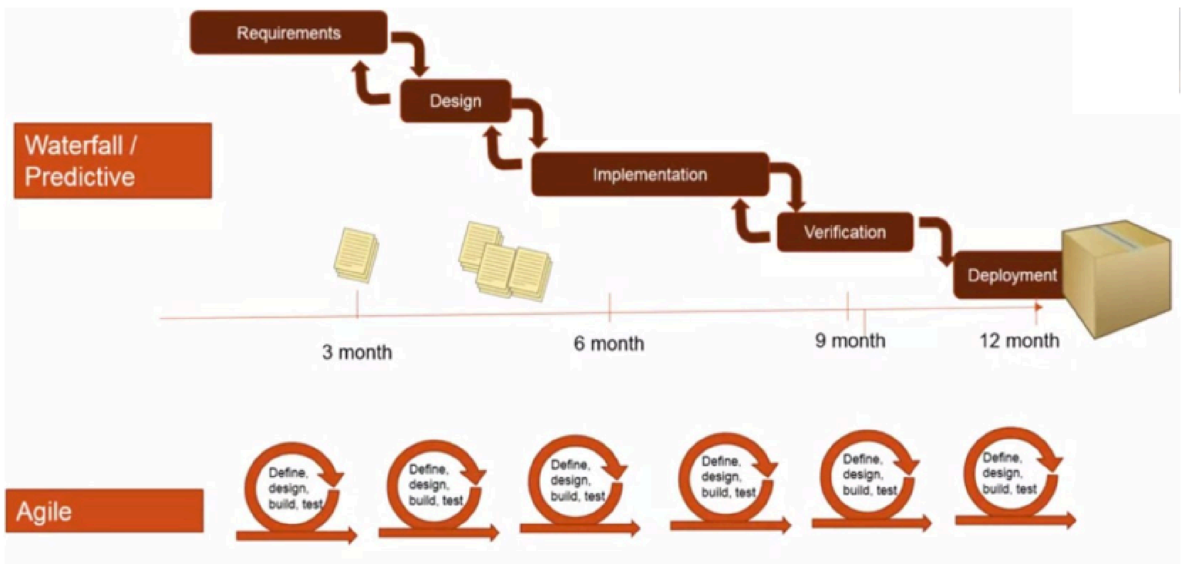
1. **Individuals and interactions:** Valuing human collaboration over rigid processes and tools.
2. **Working software:** Emphasizing the delivery of working software over comprehensive documentation.
3. **Customer collaboration:** Prioritizing close collaboration with customers throughout the development process.
4. **Responding to change:** Embracing change and adapting the system to evolving requirements.

Principles of Agile Methods

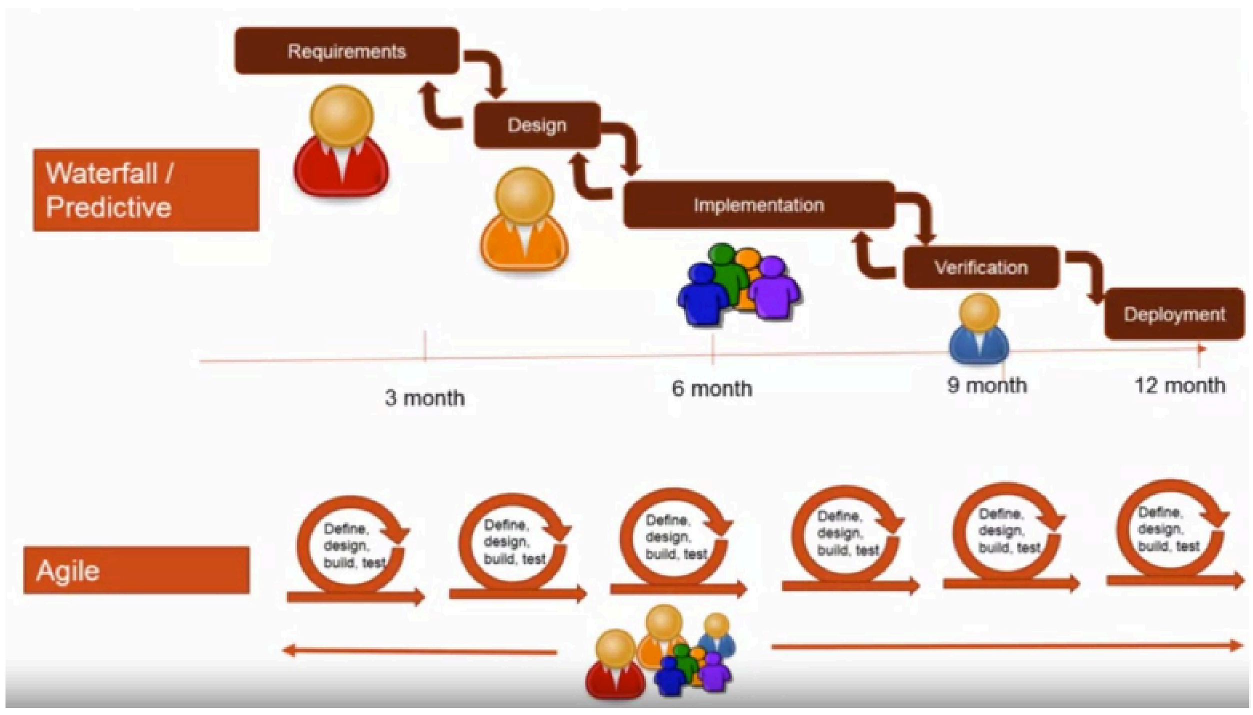
Agile methods have following principles:

1. **Incremental delivery:** Developing software in small, manageable increments.
2. **Customer involvement:** Keeping customers closely involved in requirement gathering and evaluation.
3. **People not process:** Recognizing and leveraging the skills of the development team.
4. **Embrace change:** Anticipating and accommodating changing requirements.
5. **Maintain simplicity:** Focusing on simplicity in both the software and the development process.

Big Batch Vs. Small Byte size chunks



Handoff Vs. Collaboration



Agile method applicability

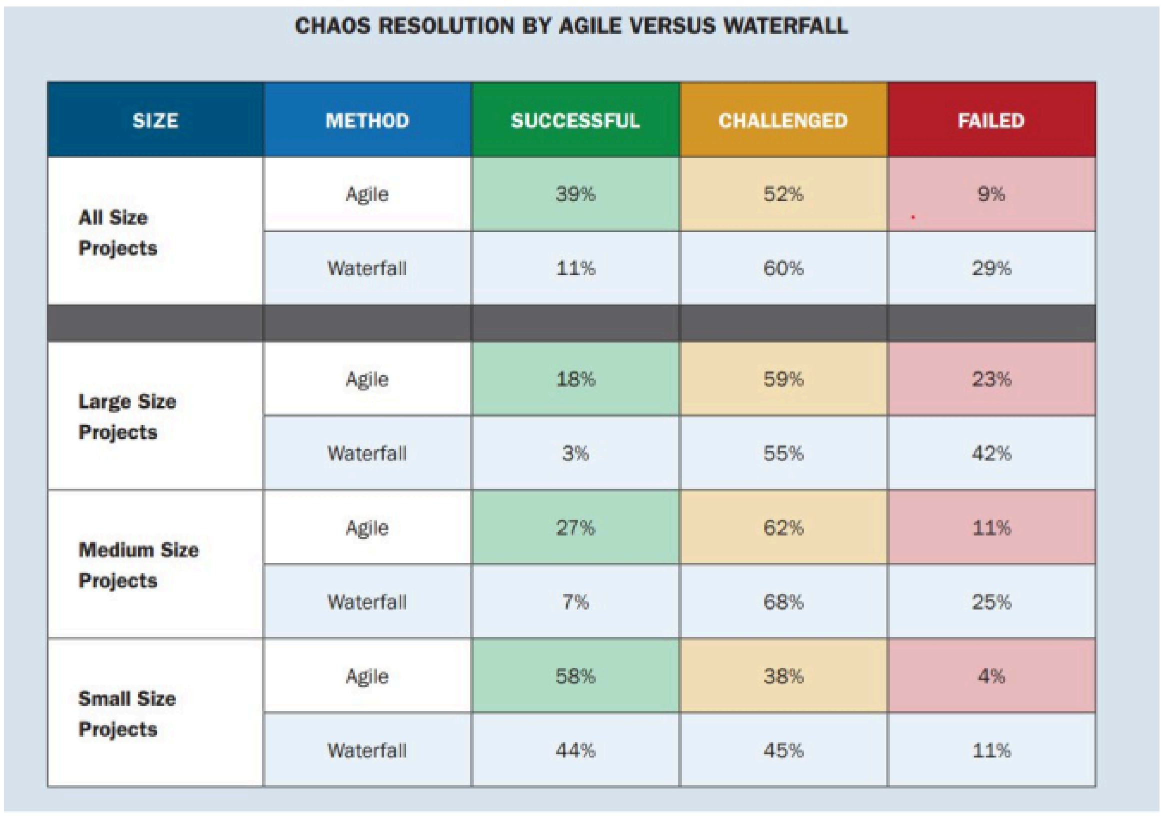
Agile methods are particularly suitable for:

- **Product development:** Software companies developing small to medium-sized products for sale.
    - ➔ Virtually all software products and apps are now developed using agile approaches.

- **Custom system development**: Within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are few external rules and regulations that affect the software.

When to Use Agile

Agile is not limited to small projects. Large organizations like Lockheed Martin, Google, Microsoft, and SpaceX use agile methods. However, agile is not suitable for all situations . In Predictive and repeatable work, Agile is less effective for tasks that require precise planning and predictable outcomes.



CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Prerequisites for Agile Success:

- **Close collaboration with businesses and users:** Agile requires active involvement from stakeholders.
- **Team readiness for change:** Agile teams must be adaptable and willing to embrace change.

Agile Frameworks

1. Extreme Programming (XP)
2. Scrum

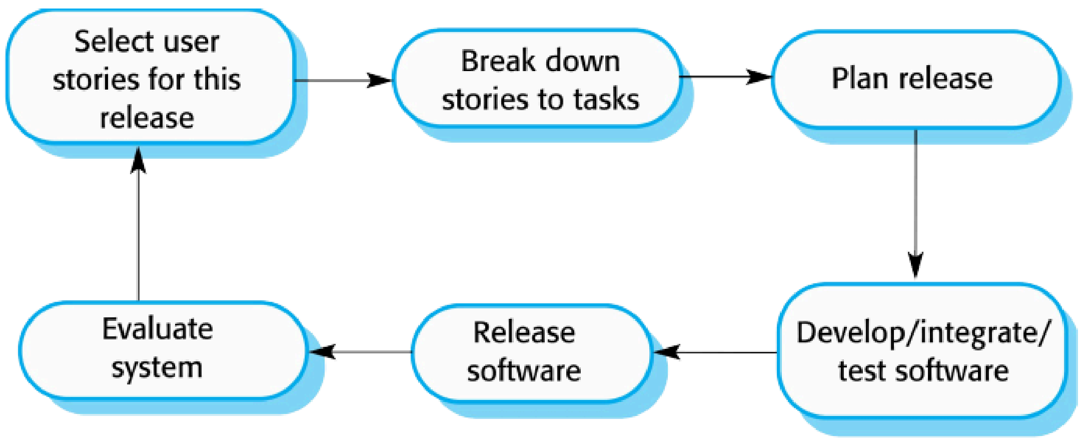
Extreme Programming (XP)

XP is an agile software development method,developed in 1996 that emphasizes iterative development, customer involvement, and continuous improvement.

Key Principles:

- **Incremental development:** Building software in small, frequent increments.
- **Customer involvement:** Close collaboration with customers throughout the development process.
- **People not process:** Valuing the skills of the development team and avoiding prescriptive processes.
- **Embrace change:** Anticipating and accommodating changing requirements.
- **Maintain simplicity:** Focusing on code simplicity and maintainability.

XP Life Cycle


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

XP and Agile Principles

XP practices align with the following agile principles:

- **Incremental development:** XP supports incremental development through frequent system releases.
- **Customer involvement:** XP emphasizes full-time customer engagement with the development team.

- **People not process:** XP promotes collective ownership, pair programming, and avoids long working hours.
- **Change:** XP facilitates change through regular system releases.
- **Simplicity:** XP maintains simplicity through constant code refactoring.

Practices

Small Releases

- Developing the minimum viable product first.
- Delivering frequent incremental updates.

Simple Design

- Designing only what is necessary to meet current requirements.

Incremental Planning

-
- Breaking requirements into manageable tasks.
- Estimating task duration and prioritizing them.

Test-First Development

- Writing automated tests before implementing functionality.

Refactoring

- Continuously improving code quality by removing duplication and complexity.

Pair Programming

- Developers working in pairs, reviewing each other's work.

Collective Ownership

- All developers responsible for all code.

Continuous Integration

- Integrating new code into the main branch frequently.

Sustainable Pace

- Avoiding excessive overtime to maintain code quality and productivity.

On-Site Customer

- A customer representative actively involved in the development process.

Influential Practices

XP has a strong technical focus and can be challenging to integrate with management practices in many organizations. This is because XP emphasizes:

- **Customer involvement:** Customers are actively involved in the development process, which may not align with traditional management structures.
- **Team autonomy:** XP teams have a high degree of autonomy, which can conflict with hierarchical management styles.
- **Iterative development:** XP focuses on delivering working software in short iterations, which may not fit into traditional project planning and budgeting cycles.

Key Practices in Agile Development

While XP is not widely used in its original form, several of its key practices have been adopted by agile development methodologies:

1. **User stories:** Describing requirements as scenarios that are easy for customers to understand.

Example: As a customer, I want to be able to search for products so that I can find the one I need quickly and easily.

2. **Refactoring:** Continuously improving code quality by removing duplication and complexity.

Example: Reorganization of a class hierarchy to remove duplicate code.

3. **Test-first development:** Writing automated tests before implementing functionality to ensure code correctness.

4. **Pair programming:** Developers working together to improve code quality and knowledge sharing.

These practices have been integrated into agile methodologies to improve software development efficiency and quality, while addressing some of the challenges associated with XP's strict adherence to its principles.

Customer Involvement

Customers play a crucial role in XP by:

- Defining requirements.
- Developing acceptance tests.
- Validating new code.

Test Automation

XP emphasizes test automation to:

- Ensure code quality through frequent testing.
- Catch errors introduced by new functionality.

Problems with Test-First Development

Test-first development can face challenges such as:

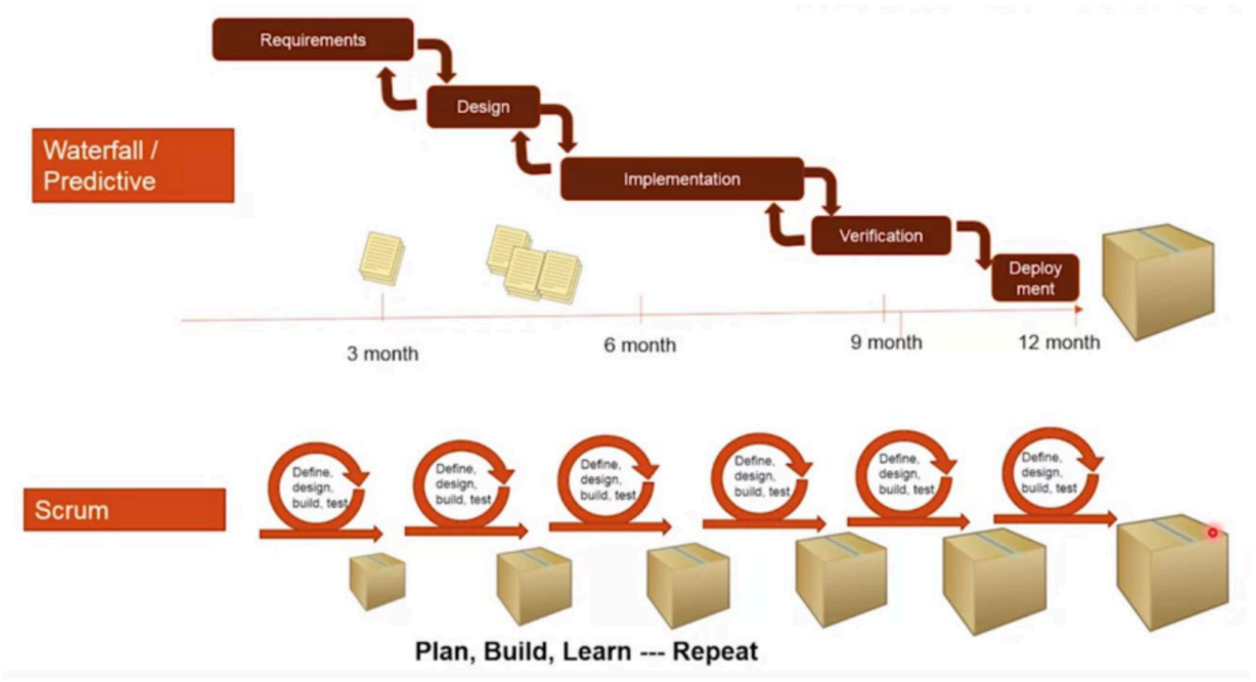
- Incomplete or inaccurate tests due to programmer shortcuts.
- Difficulty writing tests for complex user interfaces.
- Judging the completeness of test coverage.

Scrum

Scrum is an agile software development framework that focuses on managing iterative development through a series of sprints.

Phases in Scrum

1. **Planning:** Establishing project objectives and designing the software architecture.
2. **Sprint Cycles:** Iterative development cycles where each cycle produces a potentially shippable product increment.
3. **Closure:** Wrapping up the project, completing documentation, and assessing lessons learned.



Terminology

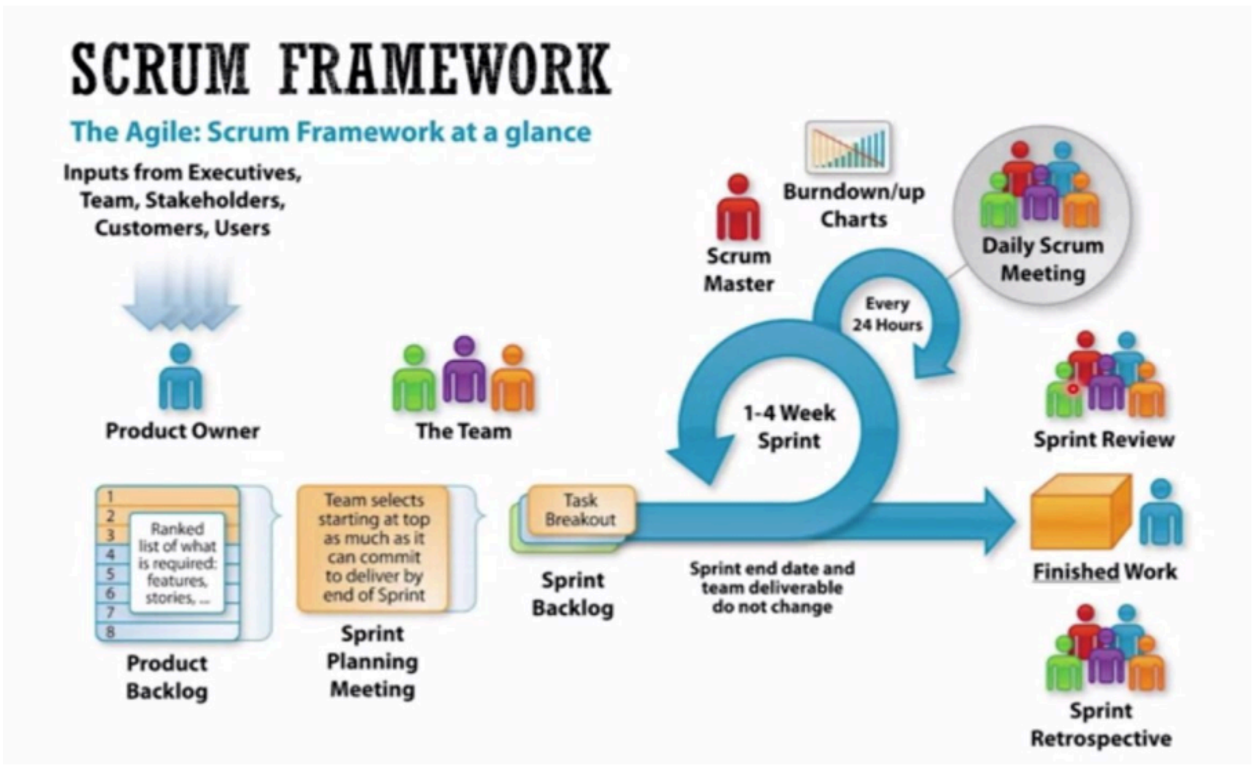
- **Development Team:** Self-organizing group of developers responsible for software development.
- **Potentially Shippable Product Increment:** Deliverable from each sprint that is potentially ready for release.
- **Product Backlog:** List of prioritized items to be developed.
- **Product Owner:** Individual responsible for defining and prioritizing product features.
- **Scrum Master:** Facilitator who ensures Scrum process is followed and protects the team from distractions.
- **Sprint:** Fixed-length development iteration (usually 2-4 weeks).
- **Velocity:** Estimate of the team's capacity to complete product backlog items in a sprint.

The Scrum Sprint Cycle

- **Selection:** Team selects features from the product backlog to be developed in the sprint.
- **Development:** Team develops the software, isolated from external interference.
- **Review:** Work is presented to stakeholders at the end of the sprint.

Teamwork in Scrum

- **Scrum Master:** Facilitates daily meetings, tracks progress, and communicates with stakeholders.
- **Daily Scrums:** Short meetings where team members share progress and plan for the next day.



SCRUM FRAMEWORK
The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

Benefits of Scrum

- Manageable and understandable product chunks.
- Progress is not hindered by unstable requirements.
- Improved team communication and visibility.
- On-time delivery of increments and customer feedback.
- Trust and a positive project culture.