# 1.1.2 Agile Basics Summary Points

courses.edx.org/courses/course-v1:USMx+ENCE607.1x+3T2019/courseware/eacf1c65220b48bdaa22de2734fc2bf5/ce31807d48ea4bf58066d7bf48df062d

Agile Manifesto was codified in 2001 in Snowbird by Scrum, XP, and DSDM practitioners. Agile Manifesto includes:

- Individuals and Interactions OVER processes and tools
- Working software (systems) OVER comprehensive documentation
- Customer collaboration OVER contract negotiation
- Responding to change OVER following a plan

These values are at the core of why agile works and continues to be used on projects with high uncertainty today.

Sprint basics include the three parts of a Sprint:

- Sprint Planning
- Sprint Development
- Sprint Retro & Review

A Sprint is a timebox, or a period that is used to contain the time allowed for work to be completed. It can be anywhere from two weeks to a month (although shorter is becoming more popular among advanced practitioners).

Sprint Planning starts with the Product Owner selecting the work to be done from the Product's Backlog. The Product Backlog is the list of work that is prioritized by its importance, either for ongoing improvement or the completion of a new product. The Team reviews the stories and then selects what work they will be able to complete during the sprint. This process is facilitated by the Scrum Master who does not participate in the doing of work, but instead is focuses on enabling the team to move quickly with good processes and best practices. The final set of stories should form a cohesive "product increment" that the team can demonstrate by the end of the sprint.

- Input: Product Backlog of stories prioritized by the Product Owner
- Process: Review and select stories for the sprint
- Output: Sprint Backlog of stories the team commits to complete by the end of the Sprint

Sprint Development begins with the daily standup. Daily standups are self-reporting of the team on what work they will get done that day. This is usually done around a Kanban board, or other big visual information radiator (BVIR). The team opens only a few stories at a time

and work story-by-story to analyze, build, and test the work. The result by the end of the Sprint is a shippable increment that can be demonstrated to the product owner. Meetings are facilitated by the Scrum Master, and the Product Owner determines if a Story is complete to meet the stakeholder needs.

- Input: Sprint Backlog of stories the team commits to complete by the end of the Sprint
- Process: Daily reporting and execution against a few stories at a time: designing, building, testing and closing
- Output: A shippable product increment that can be demonstrated

Sprint Retro and Review are ceremonies to gain feedback and drive continuous improvement into the team. The first step is the Sprint Review, where the Product Owner demonstrates the product increment from the Sprint to Stakeholders. This is an opportunity to gain stakeholder buy-in and feedback, so the team knows its on track with the product direction. The Product Owner can also get feedback on what should be in the next Sprint. The Sprint Retrospective or "Retro" is the second ceremony used to close a sprint. The Retro involves the team going into a room to evaluate how the sprint went, and identify opportunities for improvement in the next sprint. The best Sprint Retros are run as games to facilitate input from the whole team and quickly identify improvements.

- Input: Shippable product increment that can be demonstrated
- Process: Demonstrations and games to facilitate feedback on the product and team processes
- Output: Feedback on the product's direction and actions to improve the next sprint

Iron Triangle helps to explain how the different project management methods align. The Iron Triangle includes:

- Scope - the technical work to be done
- Schedule - the total calendar time to execute the work
- Budget - the total cost of the project in dollars

All aspects of the Iron Triangle are constraints and costs to the organization. More schedule means a delay of project benefits and tie-up of capital. More budget means more dollars or capital invested. More scope means a larger product to support or maintain for the organization. These are all forms of cost and constrain how the work can be accomplished when they are fixed.

The three types of project management are Agile, Traditional, and Lean.

- Agile - varies scope against fixed budget and schedule
- Traditional - varies budget against fixed scope and schedule
- Lean - varies schedule (or solution time) against fixed scope and budget

The goals and requirements of each method are essential for understanding the place of each method in the project manager's arsenal:

- Agile - goal is speed (deliver early versions fast), and requires trust to minimize scope for fast value delivery
- Traditional - goal is efficiency (best price), and requires efficiency to deliver lowest cost on time and budget
- Lean - goal is to innovate (solve problems), and requires expertise to minimize time of delivery

False comparisons across types of projects abound. Many times the objections one hears about using Agile is that it's missing critical elements, such as design, testing, or documentation. These are all wrong. In fact, every project must have the following to be successful:

- Charter
- Plan
- Documentation
- Design
- Testing

Remember that we vary scope to target just what the customer needs, so we don't waste time or money in the process. That's the power of varying scope. It's fast and limits waste by reducing the work to a minimum viable product (MVP) that meets the project objectives (in the charter). To do this, every Agile project needs:

- Shared Vision Robust to Change (can vary scope and stay on target)
- Whole Teams (customer + cross-functional team)
- Incremental Delivery (learn by doing and using small "sprints")
- Continuous Integration & Testing (teams test increments to ensure they work)

Scrum, SAFe, or Disciplined Agile are all frameworks that help define roles and processes to scale and implement the methodology of Agile. They provide a shared language. But the method remains the same.