## Lesson 7

## Agile Development

**Learning Outcomes:**

After successful completion of this lesson, you should be able to:

1. Discuss the Agile process.
2. Identify the key traits of the people in an agile team.
3. Describe the different approach to agile software development.

.

**Course Materials:**

**The Development of Agile Process Model**

**The Agile Manifesto**

In 2001, Kent Beck and 16 other noted software developers, writers, and consultants (referred to as the “Agile Alliance”) signed the “Manifesto for Agile Software Development.” It stated:

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.

**Agile Software Engineering**

Agile software engineering combines a philosophy and a set of development guidelines.

The philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity.

The development guidelines stress delivery over analysis and design (although these activities are not discouraged), and active and continuous communication between developers and customers.

An agile philosophy for software engineering stresses four key issues: the importance of self-organizing teams that have control over the work they perform, communication and collaboration between team members and between practitioners and their customers, a recognition that change represents an opportunity, and an emphasis on rapid delivery of software that satisfies the customer. Agile process models have been designed to address each of these issues.

**Agile Process**

Agile software process is characterized in a manner that addresses a number of key assumptions about the majority of software projects:

1. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds.

1. For many types of software, design and construction are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to predict how much design is necessary before construction is used to prove the design.

1. Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.

**Agility Principles**

The Agile Alliance defines 12 agility principles for those who want to achieve agility:

* 1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

* 1. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

* 1. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

* 1. Business people and developers must work together daily throughout the project.

* 1. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

* 1. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

* 1. Working software is the primary measure of progress.

* 1. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

* 1. Continuous attention to technical excellence and good design enhances agility.

* 1. Simplicity—the art of maximizing the amount of work not done—is essential.

* 1. The best architectures, requirements, and designs emerge from self– organizing teams.

* 1. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

**Key Traits of the People in an Agile Team**

1. **Competence.** In an agile development (as well as software engineering) context, “competence” encompasses innate talent, specific software-related skills, and overall knowledge of the process that the team has chosen to apply. Skill and knowledge of process can and should be taught to all people who serve as agile team members.

1. **Common focus.** Although members of the agile team may perform different tasks and bring different skills to the project, all should be focused on one goal—to deliver a working software increment to the customer within the time promised. To achieve this goal, the team will also focus on continual adaptations (small and large) that will make the process fit the needs of the team.

1. **Collaboration.** Software engineering (regardless of process) is about assessing, analyzing, and using information that is communicated to the software team; creating information that will help all stakeholders understand the work of the team; and building information (computer software and relevant databases) that provides business value for the customer. To accomplish these tasks, team members must collaborate—with one another and all other stakeholders.

1. **Decision-making ability.** Any good software team (including agile teams) must be allowed the freedom to control its own destiny. This implies that the team is given autonomy—decision-making authority for both technical and project issues.

1. **Fuzzy problem-solving ability.** Software managers must recognize that the agile team will continually have to deal with ambiguity and will continually be buffeted by change. In some cases, the team must accept the fact that the problem they are solving today may not be the problem that needs to be solved tomorrow. However, lessons learned from any problem-solving activity (including those that solve the wrong problem) may be of benefit to the team later in the project.
2. **Mutual trust and respect.** The agile team must become what DeMarco and

Lister [DeM98] call a “jelled” team. A jelled team exhibits the trust and respect that are necessary to make them “so strongly knit that the whole is greater than the sum of the parts.” [DeM98]

1. **Self-organization**. In the context of agile development, self-organization implies three things: (1) the agile team organizes itself for the work to be done, (2) the team organizes the process to best accommodate its local environment, (3) the team organizes the work schedule to best achieve delivery of the software increment. Self-organization has a number of technical benefits, but more importantly, it serves to improve collaboration and boost team morale. In essence, the team serves as its own management.

**Agile Process Models**

* **Extreme Programming**

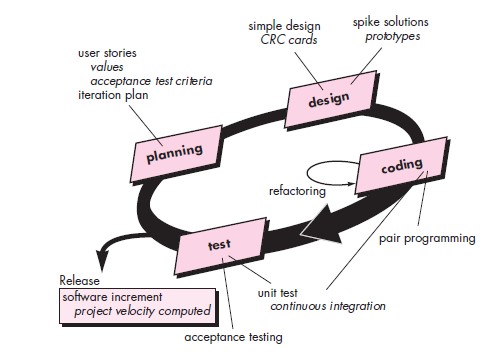
Extreme Programming (XP) is the most widely used approach to agile software development.

**Five Values that Establish Foundation for All Work Performed as Part of XP**

* 1. Communication
  2. Simplicity
  3. Feedback
  4. Courage
  5. Respect.

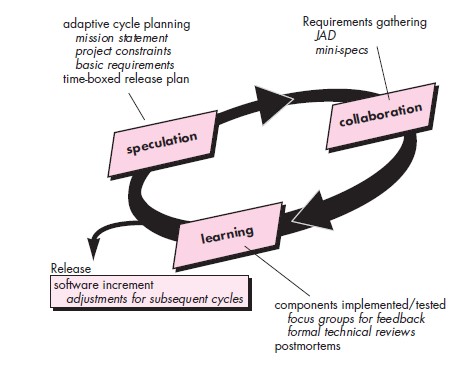
Each of these values is used as a driver for specific XP activities, actions, and tasks.

**The XP Process**



* **Adaptive Software Development (ASD)**

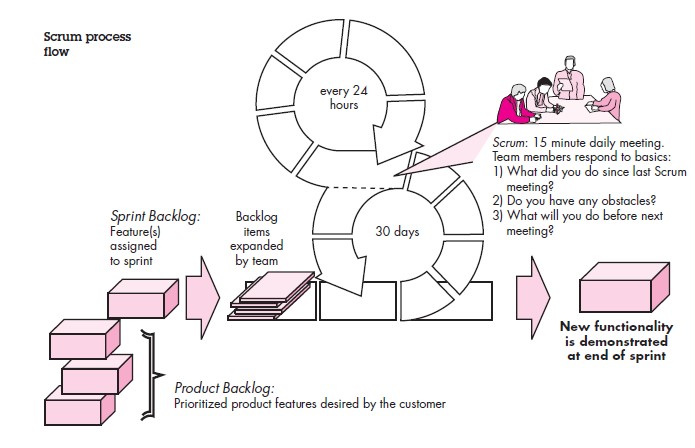
Adaptive Software Development (ASD) has been proposed by Jim Highsmith as a technique for building complex software and systems. The philosophical underpinnings of ASD focus on human collaboration and team self-organization.



* **Scrum**

Scrum (the name is derived from an activity that occurs during a rugby match) is an agile software development method that was conceived by Jeff Sutherland and his development team in the early 1990s.

Scrum principles are consistent with the agile manifesto and are used to guide development activities within a process that incorporates the following framework activities: requirements, analysis, design, evolution, and delivery.



Scrum emphasizes the use of a set of software process patterns that have proven effective for projects with tight timelines, changing requirements, and business criticality.

Each of these process patterns defines a set of development actions:

Backlog—a prioritized list of project requirements or features that provide business value for the customer. Items can be added to the backlog at any time (this is how changes are introduced). The product manager assesses the backlog and updates priorities as required.

Sprints—consist of work units that are required to achieve a requirement defined in the backlog that must be fit into a predefined time-box14 (typically 30 days).

Changes (e.g., backlog work items) are not introduced during the sprint. Hence, the sprint allows team members to work in a short-term, but stable environment.

Scrum meetings—are short (typically 15 minutes) meetings held daily by the Scrum team. Three key questions are asked and answered by all team members:

* What did you do since the last team meeting?
* What obstacles are you encountering?
* What do you plan to accomplish by the next team meeting?

A team leader, called a Scrum master, leads the meeting and assesses the responses from each person. The Scrum meeting helps the team to uncover potential problems as early as possible. Also, these daily meetings lead to “knowledge socialization” and thereby promote a self-organizing team structure.

Demos—deliver the software increment to the customer so that functionality that has been implemented can be demonstrated and evaluated by the customer. It is important to note that the demo may not contain all planned functionality, but rather those functions that can be delivered within the time-box that was established.

• **Dynamic Systems Development Method (DSDM)**

The Dynamic Systems Development Method (DSDM) is an agile software development approach that “provides a framework for building and maintaining systems which meet tight time constraints through the use of incremental prototyping in a controlled project environment”.

The consortium has defined an agile process model, called the DSDM life cycle that defines three different iterative cycles, preceded by two additional life cycle activities:

Feasibility study—establishes the basic business requirements and constraints associated with the application to be built and then assesses whether the application is a viable candidate for the DSDM process.

Business study—establishes the functional and information requirements that will allow the application to provide business value; also, defines the basic application architecture and identifies the maintainability requirements for the application.

Functional model iteration—produces a set of incremental prototypes that demonstrate functionality for the customer. (Note: All DSDM prototypes are intended to evolve into the deliverable application.) The intent during this iterative cycle is to gather additional requirements by eliciting feedback from users as they exercise the prototype.

Design and build iteration—revisits prototypes built during functional model iteration to ensure that each has been engineered in a manner that will enable it to provide operational business value for end users. In some cases, functional model iteration and design and build iteration occur concurrently.

Implementation—places the latest software increment (an “operationalized” prototype) into the operational environment. It should be noted that (1) the increment may not be 100 percent complete or (2) changes may be requested as the increment is put into place. In either case, DSDM development work continues by returning to the functional model iteration activity.

DSDM can be combined with XP to provide a combination approach that defines a solid process model (the DSDM life cycle) with the nuts and bolts practices (XP) that are required to build software increments. In addition, the ASD concepts of collaboration and self-organizing teams can be adapted to a combined process model.

• **Other Agile Process Models**

1. Crystal
2. Feature Driven Development (FDD)
3. Lean Software Development (LSD)
4. Agile Modelling (AM)
5. Agile Unified Process (AUP)

**Tool Set for Agile Process**

The “tool set” that supports agile processes focuses more on people issues than it does on technology issues.

**Read:**

Chapter 3 Software Engineering, Practitioner Approach 7th Edition Roger S. Pressman Ph.D.

**Activities/Assessment:**

**Fill-in the blanks.**

1. \_\_\_\_\_\_\_\_\_\_\_\_\_is a process framework that can adopt the tactics of another agile approach such as XP.
2. \_\_\_\_\_\_\_\_\_\_\_\_\_consist of work units that are required to achieve a requirement defined in the backlog that must be fit into a predefined time-box14 (typically 30 days).
3. \_\_\_\_\_\_\_\_\_\_\_\_\_ is the most widely used approach to agile software development.
4. \_\_\_\_\_\_\_\_\_\_\_\_\_a prioritized list of project requirements or features that provide business value for the customer.
5. \_\_\_\_\_\_\_\_\_\_\_\_\_ is the alliance that signed the “Manifesto for Agile Software Development.”