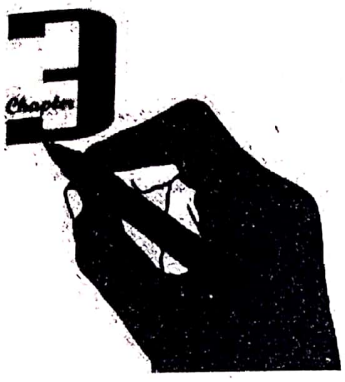



# AGILE SOFTWARE DEVELOPMENT

## CHAPTER OUTLINE

**After studying this chapter, the reader will be able to understand the**

- Agile Methods
- Agile Principles
- Extreme Programming
- Testing in XP
- Agile Project Management

## AGILE METHODS

In the 1980s and early 1990's there was a widespread view that the best way to achieve better software was through careful project planning, formalized quality assurance, the use of analysis and design methods supported by CASE tools and controlled and rigorous software development process. However, when this heavy weight plan based development approach was applied to smaller and medium sized business systems, the overhead involved was too large and then it sometimes dominated the software development process, More time was spent on how the system should be developed then on program development and testing. The conventional software models such as Waterfall Model that depends on completely specifying the requirements, designing, and testing the system are not geared towards rapid software development. As a consequence, a conventional software development model fails to deliver the required product. This is where the agile software development comes to the rescue. It was specially designed to curate the needs of the rapidly changing environment by embracing the idea of incremental development and develop the actual final product. These methods allowed the development team to focus on software itself rather than on its plan, design and documentations.

"Agile software development is a software development method based on iterative and incremental development in which requirement and solutions evolve through collaboration between self organizing, cross functional teams." It generally promotes a disciplined project management process that encourages frequent inspection and adaption, a leadership philosophy that encourages a team works self-organizing and accountability, a set of engineering best practices intended to allow for rapid delivery of high quality software. Agile software development combines the philosophy and set of set of development guidelines that encourses the customer satisfaction and early development of software together with small, highly motivated software development team minimum software engineering work product and overall development simplicity.

## AGILE PRINCIPLES

There are five major agile development principles

1. **Customer Involvement**
   In agile development, customers are closely involved in the development team throughout the development process. Their role in development team is to evaluate the newly developed increments, provide the feedback and priorities new system requirements for new increment of the system.

2. **Incremental Delivery**
In agile development, software is developed in increments with the customer specifying the requirements to be included in each increment.

3. **People not Process**

In this method, the skills of development team should be recognized and exploited then team members should be left to develop in their own ways of working without prescriptive processes.

4. **Embrace Change**

Agile development process expects the system requirement to change as much as possible so encourage the customer to change the requirement during early stage of development so that system can be designed to accommodate these changes.

5. **Maintain Simplicity**

In agile development we should focus of Simplicity in both the software being developed and the development process. Development team actively works to eliminate complexity from the system wherever possible.

Apart from above there are 12 Principles are based on the Agile Manifesto.

- Highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes connect change for the customer's competitive advantages.
- 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.

## Advantages of Agile Method

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

## Disadvantages of Agile Method

- In case of software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

## When to use Agile Method

- When new changes are needed to be implemented. Agile method provides facility such that new changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly affected or removed based on feedback. This effectively gives the customer the finished system they want or need.

## Plan-driven and Agile Development

Plan driven development is a technique where all of the process activities are planned in advance and progress is measured against this plan. This approach to software development identifies separate stages in the software process with outputs associated with each stage. The outputs from one stage are used as a basis for planning the following process activity. In a plan driven approach, iteration occurs within activities with formal documents used to communicate between stages of the process. It is perfectly feasible to allocate requirements and plan the design and development phase as a series of increments.

In an agile approach, planning is incremental and it is easier to change the plan and software to reflect changing customer requirements. In this iteration occurs across activities. Therefore, the requirements and the design are developed together, rather than separately. It is perfectly feasible to allocate requirements and plan the design and development phase as a series of increments. An agile process is not inevitably code-focused and it may produce some design documentation. The decision on whether to use plan driven approach or an agile to development should depend on the type of software being developed, the capabilities of the development team and the culture of the company developing the system. In practice, most practical processes include elements of both plan driven and agile approaches.

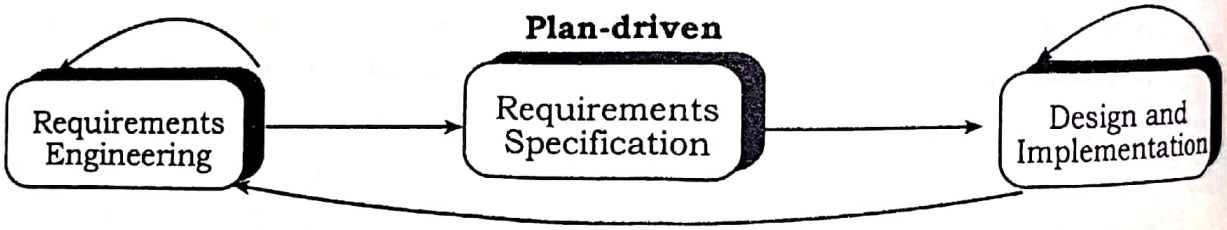



**Fig: Plan-driven Development**

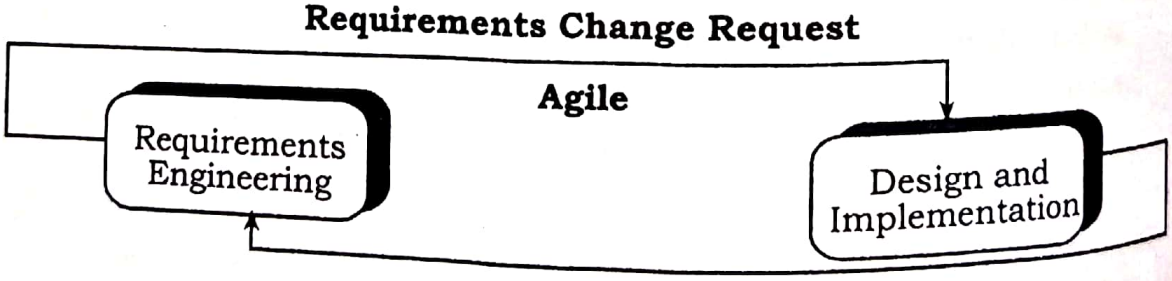



**Fig: Agile Development**

While developing the software system, to decide the balance between plan driven and agile approach, we should focus on following facts:

- If you have a very detailed specification and design before moving to implementation then, software engineer needs to use a plan-driven approach.
- If software engineer have to deliver the software to customers using an incremental delivery strategy and get rapid feedback from them, use agile methods.

- Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.
- Plan-driven approaches may be required for systems that require a lot of analysis before implementation (e.g. real-time system with complex timing requirements).
- Long-lifetime systems may require more design documentation to communicate the original intentions of the system developers to the support team. Use plan driven approach.
- It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code
- If the development team is distributed or if part of the development is being outsourced, then software engineer may need to develop design documents to communicate across the development teams. Then you need to use plan driven approach.

## EXTREME PROGRAMMING

Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent releases in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted. It is the discipline of software development based on values of simplicity, communication, feedback, and courage. Other elements of Extreme Programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels.

XP attempts to reduce the cost of changes in requirements by having multiple short development cycles, rather than a long one. On this basis, changes are a natural, unavoidable and desirable aspect of software-development projects, and should be planned for, instead of attempting to define a non-variable set of requirements.

Extreme programming also introduces a number of basic values, principles and practices on top of the agile programming framework.

Extreme Programming initially recognizes five values:

- Communication
- Simplicity
- Feedback
- Courage
- Respect

## Communication

Building software systems requires communicating system requirements to the developers of the system. In formal software development methodologies, this task is accomplished through documentation. Extreme programming techniques can be considered as methods for rapid developments and sharing best practices and knowledge among members of a development team. The goal is to give all developers a shared view of the system which matches the user's view of the system. For this reason extreme programming favors simple designs, collaboration of users and programmers, frequent verbal communication, and feedback.

## Simplicity

Extreme programming encourages starting with the simplest solution. Extra functionality can then be added later. Coding and designing for uncertain future requirements implies the risk of spending resources on something that might not be needed, while perhaps delaying crucial features. Related to the communication" value, simplicity in design and coding should improve the quality of communication. A simple design with very simple code could be easily understood by most programmers in the team.

## Feedback

Feedback is closely related to communication and simplicity. Flaws in the system are easily communicated by writing a unit test that proves a certain piece of code will break. The direct feedback from the system tells programmers to recode this part. Within extreme programming, feedback relates to different dimensions of the system development:

- Feedback from the system: by writing unit tests, or running periodic integration tests, the programmers have direct feedback from the state of the system after implementing changes.
- Feedback from the customer: The functional tests are written by the customer and the testers. They will get concrete feedback about the current state of their system. This review is planned once in every two or three weeks so the customer can easily steer the development.
- Feedback from the team: When customers come up with new requirements in the planning game the team directly gives an estimation of the time that it will take to implement.

## Courage

Courage enables the developers to feel comfortable with refactoring their code when necessary. This means reviewing the existing system and modifying it so that future changes can be implemented more easily. Courage also means persistence: A programmer might be stuck on a complex problem for an entire day, then solve the problem quickly the next day, but only if they are persistent.

## Respect

The respect value emphasizes on respect for others as well as self-respect. Programmers should never commit changes that break compilation, that make existing unit-tests fail, or that otherwise delay the work of their peers.

Adopting the four earlier values leads to respect gained from others in the team. Nobody on the team should feel unappreciated or ignored. This ensures a high level of motivation and encourages loyalty toward the team and toward the goal of the project. This value is very dependent upon the other values, and is very much oriented toward people in a team.

Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development "Tasks". Developers work in pairs, checking each other's work and providing the support to always do a good job. An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented. XP is a s/w development methodology which is intended to improve s/w quality and responsiveness to changing customer requirements.

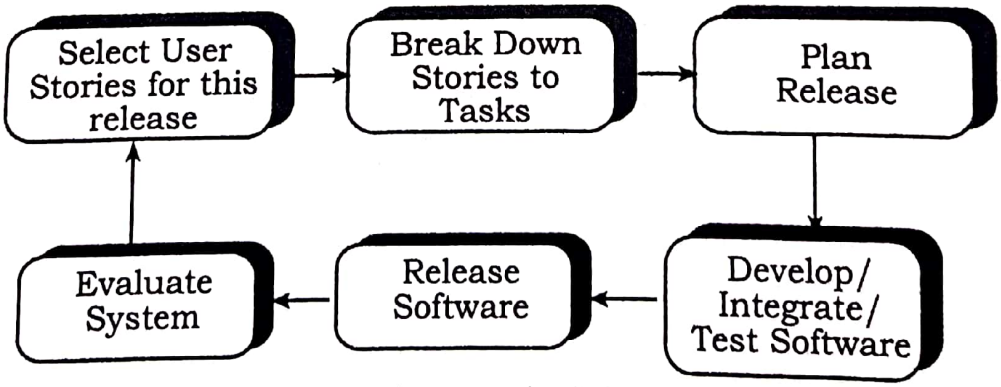



**Fig: Extreme Programming Release cycle**

Extreme programming support principles of agile as follows:

- Incremental development is supported through small, frequent releases of the software and by an approach to requirements description based on customer stories as scenarios that can be basis for deciding next increment.

- Customer's involvement is supported through the full-time engagement of the customer in the development team. The customer representative takes part in the development and is responsible for defining acceptance tests for the system.
- People, not process, are supported through pair programming, collective ownership of the system code, and a sustainable development process that does not involve excessively long working hours.
- Change is supported through regular system release to customer, test first development and continuous integration of new features.
- Maintaining simplicity is supported through constant refactoring to improve code quality and by using simple design that do not anticipate future changes to the system.

| Principle or practice | Description |
|---|---|
| Incremental planning | Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development "Tasks". |
| Small releases | The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release. |
| Simple design | Enough design is carried out to meet the current requirements and no more. |
| Test-first development | An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented. |
| Refactoring | All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable. |
| Pair programming | Developers work in pairs, checking each other's work and providing the support to always do a good job. |
| Collective ownership | The pairs of developers work on all areas of the system and all the developers own all the code. Anyone can change anything. |
| Continuous integration | As soon as work on a task is complete it is integrated into the whole system. After any such integration, all the unit tests in the system must pass. |
| Sustainable pace | Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium-term productivity |
| On-site customer | A representative of the end-user of the system should be available full time for the use of the XP team. |

## TESTING IN XP

In the iterative development processes there is no system specification that can be used by an external testing team to develop system tests. As a consequence, some approaches to iterative development have only a very informal testing process. For this reason XP places more emphasis than other agile methods on the testing process. The key features of testing in XP are:

- Test-first development.
- Incremental test development from scenarios.
- User involvement in the test development and validation.
- The use of automated test cases.

Test-first development is one of the most important characteristic in XP. Writing tests first implicitly defines both an interface and a specification of behavior for the functionality being developed. User requirements in XP are expressed as scenarios or stories and the customer prioritizes these for development. The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system. Acceptance testing is the process where the system is tested using customer data to check that it meets the customer's needs.

In the test-first development the test is written before the code. More precisely, the test is written as an executable component before the task is implemented. Once the software has been implemented, the test can be executed immediately. The testing component simulates the submission of input to be tested and check that the result meets the output specification.

The automated test harness is a system that submits these automated tests for execution.

Using test-first development, there is always a set of tests that can be quickly and easily executed. This means that whenever any functionality is added to the system, the tests can be run and problems that the new code has introduced can be recognized immediately.

## Pair Programming

Another innovative practice that has been introduced is that programmers work in pairs to develop the software. They actually sit together at the same workstation to develop the software. Development does not always involve the same pair of people working together. Rather, the idea is that pairs are created dynamically so that all team members may work with other members in a programming pair during the development process.

**The use of pair programming has following advantages:**

- **Collective code ownership:** When everyone on a project is Pair Programming, and pairs rotate frequently, everybody gains a working knowledge of the entire codebase.

- **Better code:** Pair programming acts as an informal review process because each line of code is looked at by at least two people. Code inspections and reviews are very successful in discovering a high percentage of software errors.
- **Increased discipline:** Pairing partners are more likely to "do the right thing" and are less likely to take long breaks.
- **Resilient flow:** Pairing leads to a different kind of flow than programming alone, but it does lead to flow. Pairing flow is more resilient to interruptions: one programmer deals with the interruption while the other keeps working.
- **Improved moral:** Pair programming, done well, is much more enjoyable than programming alone, done well.
- **Mentoring:** Everyone, even junior programmers, has knowledge that others don't. Pair programming is a painless way of spreading that knowledge.
- **Team cohesion:** People get to know each other more quickly when pair programming.
- **Fewer interruptions:** People are more reluctant to interrupt a pair than they are to interrupt someone working alone.
- One less workstation.

## AGILE PROJECT MANAGEMENT

Software project manager have to manage the software project so that the software is delivered on time and with the estimated budget for the software. To accomplish this, manage have to follow the plan based approach and should have a stable view of everything that has to be developed and the development processes. But this does not work well with agile methods where requirement are developed incrementally, and delivered in short time interval. Agile project management is an iterative approach to delivering a project throughout its life cycle. Iterative or agile life cycles are composed of several iterations or incremental steps towards the completion of a project. Iterative approaches are frequently used in software development projects to promote velocity and adaptability since the benefit of iteration is that you can adjust as you go along rather than following a linear path. At the core, agile projects should exhibit central values and behaviors of trust, flexibility, empowerment and collaboration. To manage agile project, different approach to project management is adapted that support incremental development and strength of agile method which is realized in scrum.

## Scrum

Scrum is an iterative and incremental agile software development framework for managing agile software projects and product or application development. The concept is originated from the game named rugby football, in which scrum refers to the manner of restarting the game after a minor infraction. It refers to the agile software development model based on multiple small teams working in an intensive and interdependent manner. It is a way for team to work together to develop a product. Product development, using scrum, occurs in small pieces, with each piece building up on previously created pieces. Building products one small piece at a time encourages creativity and enables teams to respond to feedback and change, to build exactly and only what is needed.

Scrum enables teams to self-organize by encouraging physical co-location of all team members and daily face to face communication among all team members and disciplines in the project. A key principle of Scrum is its recognition that during a project the customers can change their minds about what they want and need, and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. Scrum focuses on maximizing the team's ability to deliver quickly and respond to emerging requirements.

The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

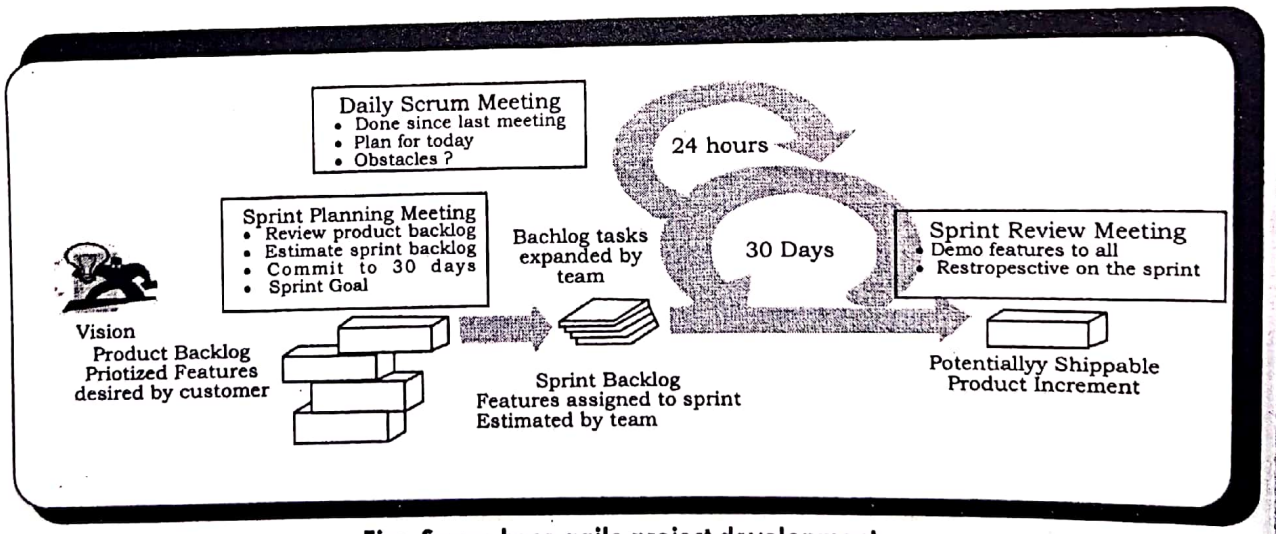



**Fig: Scrum base agile project development**

## Scrum Roles

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team.

Scrum Teams deliver products iteratively and incrementally, maximizing opportunities for feedback. Incremental deliveries of "Done" product ensure a potentially useful version of working product is always available.

### Product Owner

The Product Owner is responsible for maximizing the value of the product and the work of the Development Team and for managing the Product Backlog. Product Backlog management includes: ordering the items in the Product Backlog to best achieve goals and missions; ensuring that the Product Backlog is visible, transparent, and clear to all; ensuring the Development Team understands items in the Product Backlog to the level needed.

The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a backlog item's priority must convince the Product Owner. For the Product Owner to succeed, the entire organization must respect his or her decisions.

### Development Team

The Development Team consists of 3-9 professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint. Only members of the Development Team create the Increment. Development Teams have the following main characteristics:

- They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality;
- Development Teams are cross-functional, with all of the skills as a team necessary to create a product Increment;
- Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.

### Scrum Master

The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules. The Scrum Master is a servant-leader for the Scrum Team.

The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

### Scrum Artifacts

Scrum's artifacts represent work or value in various ways that are useful in providing transparency and opportunities for inspection and adaptation. Artifacts defined by Scrum are specifically designed to maximize transparency of key information needed to ensure Scrum Teams are successful in delivering a "Done" Increment.

### Product Backlog

The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

A Product Backlog is never complete. The earliest development of it only lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used evolves.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases. Product Backlog items have the attributes of a description, order, and estimate.

As a product is used and gains value, and the marketplace provides feedback, the Product Backlog becomes a larger and more exhaustive list. Requirements never stop changing, so a Product Backlog is a living artifact. Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog.

### Sprint backlog

Sprint Backlog is the subset of Product Backlog items selected for the Sprint together with a plan for delivering the product Increment and realizing the Sprint Goal. It is a forecast by the development team about what functionality will be in the next Increment and the work needed to deliver that functionality.

As new work is required, the development team adds it to the sprint backlog. As work is performed or completed, the estimated remaining work is updated. Only the development team can change its sprint backlog during a Sprint. It is a highly visible, real-time picture of the work that the development team plans to accomplish during the Sprint, and it belongs solely to the development team.

### Increment

The Increment is the sum of all the Product Backlog items completed during a Sprint and all previous Sprints. At the end of a Sprint, the new Increment must be "Done," which means it must be in useable condition and meet the Scrum Team's Definition of "Done." It must be in useable condition regardless of whether the Product Owner decides to actually release it.

### Scrum Events

Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. Scrum uses time-boxed events, such that every event has a maximum duration. This ensures an appropriate amount of time is spent planning without allowing waste in the planning process.

Other than the Sprint itself, which is a container for all other events, each event in Scrum is a formal opportunity to inspect and adapt something. These events are specifically designed to enable critical transparency and inspection. Failure to include any of these events results in reduced transparency and is a lost opportunity to inspect and adapt.

## The Sprint

The Sprint is a time-box of one month or less during which a "Done", useable, and potentially releasable product Increment is created. The duration of Sprint is fixed in advance for each sprint and is normally between one week and one month, although two weeks is typical. Each Sprint has a definition of what is to be built, a design and flexible plan that will guide building it, the work, and the resultant product. Each sprint is started by a planning meeting, where the tasks for the sprint are identified and an estimated commitment for the sprint goal is made, and ended by a sprint review.

Scrum emphasizes working product at the end of the Sprint that is really "done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.

## Sprint Planning Meeting

The work to be performed in the Sprint is planned at the Sprint Planning Meeting. This plan is created by the collaborative work of the entire Scrum Team.

In this part, the Development Team works to forecast the functionality that will be developed during the Sprint. The Product Owner presents ordered Product Backlog items to the Development Team and the entire Scrum Team collaborates on understanding the work of the Sprint.

The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team. By the end of the Sprint Planning Meeting, the Development Team should be able to explain to the Product Owner and Scrum Master how it intends to work as a self-organizing team to accomplish the Sprint Goal and create the anticipated Increment.

## Daily Scrum

The Daily Scrum is a 15-minute time-boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours. The Daily Scrum is held at the same time and places each day to reduce complexity. During the meeting, each Development Team member explains:

- What has been accomplished since the last meeting?
- What will be done before the next meeting?
- What obstacles are in the way?

The Development Team uses the Daily Scrum to assess progress toward the Sprint Goal and to assess how progress is trending toward completing the work in the Sprint Backlog. Daily Scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve the Development Team's level of project knowledge.

### Sprint Review

This is an informal meeting, and the presentation of the Increment. A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint. Based on that and any changes to the Product Backlog during the Sprint, attendees collaborate on the next things that could be done. The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.

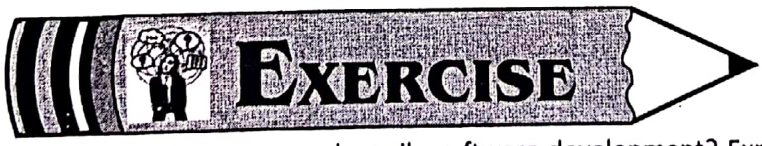
## Scaling Agile Methods

Agile methods were specially developed for the development of small and medium sized systems where there is small co-located development team and all the members work together in the same place. Agile methodologies are becoming increasingly popular for software development. However, the most popular agile methodology, called Scrum, is best suited for a single, five- to nine-person development team. In order to adopt agile processes for larger organizations, longer projects, and more complex environments, scaling is necessary.

There are two perspectives on the scaling of agile methods

a. **Scaling up Perspective:** This method of scaling agile concern with using these methods for developing large software system that cannot be developed by a small team. In this method the critical adaptation that has to be introduced is: for large system developments, it is not possible to focus only on the code of the system so, software architecture has to be designed and documents have to be produced to describe critical aspect of system such as database schema. Cross team communication mechanism has to be designed and used where teams update each other on progress.

b. **Scaling Out Perspective:** In this method major concern is with how agile methods can be introduced across a large organization with many years of software development experience.




1. What do you mean by agile software development? Explain how the principles of agile method accelerate the development of software application.
2. Differentiate between plan driven development and agile development with diagram.
3. Define extreme programming. How does extreme programming support the principles of agile development?
4. How the concept of test first development helps the programmer to develop a better understanding of the system requirements? Explain.
5. Explain the concept of pair programming and constant refactoring with their advantages.
6. What is agile project development? Explain different perspective on scaling agile methods.
7. What is scrum? Explain scrum roles and terminologies with block diagram.

□□□