CHAPTER sixteen

Testing Agile-based Software

16.1 Agile Software Development

Objectives

After reading this chapter, you should be able to understand:

- Principles of agile software development
- Agile life cycle
- Scrum method
- Agile testing life cycle
- Agile testing issues and challenges

There are various software development cycles available to develop the intended software products, such as linear sequential model, prototyping model, iterative and incremental model, spiral model, and many more. These traditional models follow the static working strategy in which everything is fixed, for example, cost of the project, requirements or scope of the project, and the schedule to be followed during development of the project. Moreover, the duration of the project is in years. Each one of these models has one or more drawbacks.

Agile software development (ASD) [176, 177, 178, 179, 180] is an amalgamation of the best practices which may be implemented by team members using Agile core values such as commitment, communication, coordination, continuous learning, and continuous improvement. It supports a unique evolution of *responding to change* from *sticking to a fixed plan* as it inherits the features of iterative and incremental model [181] along with the adaptability and predictive feature. ASD is based on lightweight methodologies (less documentation and less planning) having dynamic nature. This dynamic nature of Agile is its major strength and its nimbleness is reflected in each and every activity performed before the iteration, during the iteration, and after the iteration. In short, Agile follows a 'less is more' approach having lesser number of requirements, lesser team members in a team, smaller iterations/sprints, smaller duration, lesser documentation, and smaller meetings, among others. This leads to improved quality of software products delivered to the customer. Many software organizations have started using Agile principles and practices for the purpose of attaining higher quality in their deliverables. Largely, organizations are accepting this model as it provides an environment to accommodate frequent changes which are as per market standards and customer needs. Thus, ASD is an iterative and incremental development method and its basic concept is customer-centred. It acknowledges that requirements can change. ASD offers a professional approach to software development that encompasses human, organizational, and technological aspects of software development processes leading to a powerful way to deliver high-quality software on time.

The core ASD principles are:

- The uppermost priority should be customer approval and satisfaction by quick delivery of working software.
- Developers should welcome changing requirements, even when they are behind schedule in development.
- Working software should be delivered frequently in short release cycle of weeks rather than months.
- The most vital measure of ASD is working software.
- Daily meetings should be held between customers, business analysts and developers and face-to-face meetings between all the developers.
- ASD teams should be self-driven, resulting in self-discipline; they must also be self-motivated.
- Team should adapt to changing requirements and business needs. Team members should reflect on how to turn out to be more productive, then adjust their behavior accordingly.

16.2 Agile Model

The Agile model lays emphasis on the fact that the whole team should be a tightly integrated unit composed of developers, quality assurance members, testers, the project owner, and the customer. The key feature of ASD is to have effective communication among all team members. For valuable communication and information exchange, daily meetings are held in ASD.

Another important feature of the Agile process is iterative delivery. An iteration or delivery cycle in ASD ranges from one week to four weeks.

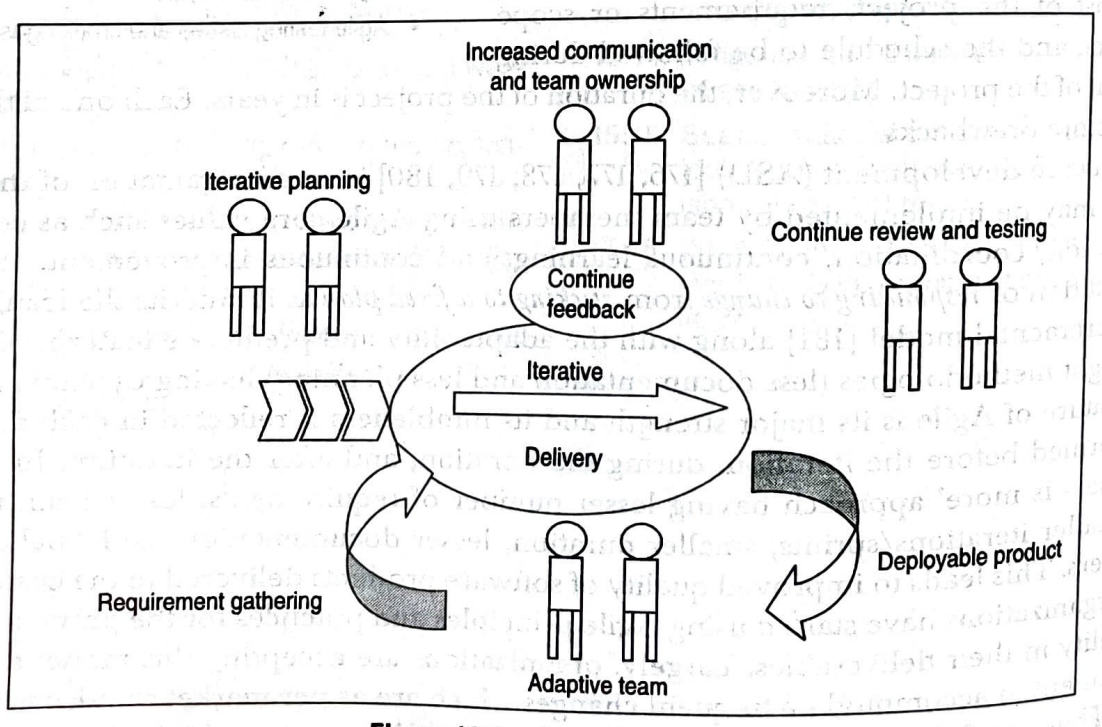

Figure 16.1 Agile processes

Agile methods break tasks into small increments with minimal planning. Iterations typically last from one to four weeks, and each iteration is worked on by a team comprising a designer, tester, sprint master, etc., and is developed using a full software development cycle including initial planning, requirements gathering and analysis, architectural and system design, coding, unit, integration, and

system and acceptance testing. This allows the project to accept changes even in later stages of development and also helps in minimizing risks. Excessive documentation is not needed; however, it can be done if required.

In ASD the emphasis is on collaboration among all stakeholders so as to improve quality in product during the sprint. The major stakeholders are the *customers*, including a direct user, requirement provider, indirect user, head of users, marketing analysts, investors, and a *development team* including the program manager, developers working on other systems that need to integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project, business people, and testers [182].

These stakeholders are the main active players in Agile working model as shown in Figure 16.1. A customer's role is to provide requirement and feedback on a timely basis and the team's role is to give a response to the feedback and apply relevant technology and market standards on a timely basis.

16.3 Agile Software Development Life Cycle

The main methods that fall under the Agile umbrella include XP, Scrum, Crystal, and DSDM. Agile methods break tasks into small increments with minimal planning, and do not directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks. Each iteration is worked on by a team through a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders. This helps minimize overall risk, and allows the project to adapt to changes quickly. Stakeholders produce documentation as required. Iterations may not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration. Multiple iterations may be required to release a product or new features as shown in Figure 16.2.

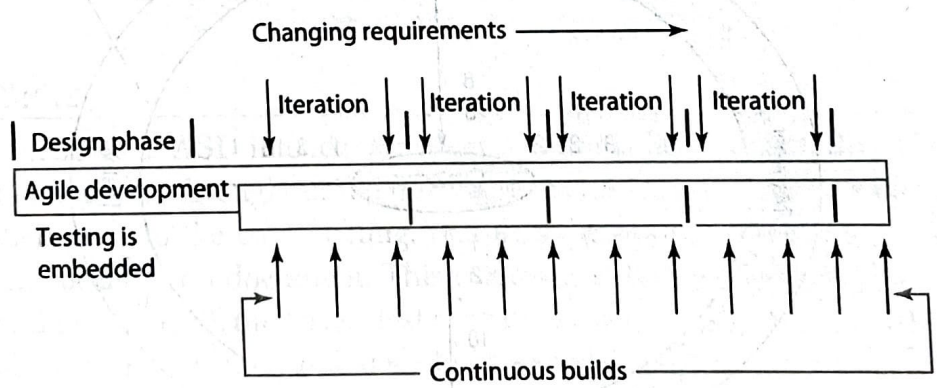

Figure 16.2 Agile life cycle

Agile SDLC includes specific activities performed by the manager (M), developers (D), testers (T), marketing professional (MP), and customer (C). Table 16.1 lists the activities performed by the different stakeholders.

At the time of production of the code or before producing the code, testing is applied by writing failed test cases unlike the traditional approach of working. Testing activity begins as soon as user stories (requirements) are finalized and prioritized, and testers try to move business logic into lower levels in order to test with lower effort in the last stage. In Agile, a quality (Q) product is delivered by operational teams, and acceptance factor (A) is related to the rate at which customer accepts the

Table 16.1 Actor activity chart

S. No.	Actor	Activity
1	C, MP, T	Requirements gathering
2	M, MP, C	Effort estimation [28], cost, risk, capacity, and resource estimation
3	M, D, T, C	User stories prioritization
4	T	Designing of test cases
5	D	Coding for the user story in the iteration
6	T	Feedback
7	D	Refactoring for the user story
8	C, M, D, T	Review meeting with demonstration
9	D, T, M, MP	Lesson learning phase or retrospective session after the iteration
10	C, MP	Release

delivered product. Effectiveness (E) of the team is related to these two factors (see Eq. 16.1). Out of these two factors, quality is more significant as acceptance is totally dependent on Q. Q is more when there is lesser number of backlogs and it is less when backlog items are more as shown in Fig. 16.3.

Backlogs can be decreased when automation is the preferred approach over a manual way of testing.

$$E = Q \times A \tag{16.1}$$

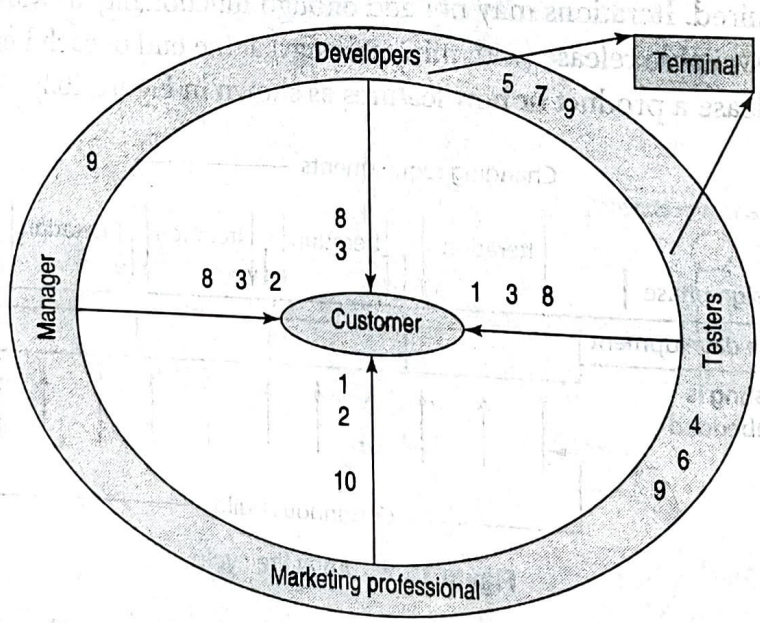

Figure 16.3 Stakeholders in Agile life cycle

16.4 Scrum

Scrum is an Agile framework which concentrates on how team members should function to produce system flexibly in a constantly changing environment. Scrum is a lightweight method to iteratively deliver software in which changes are always welcome. Scrum is concerned with the product owner, project lead, and the team working together in an intensive and interdependent manner. Scrum process

includes three phases: Pre-game, Development, and Post-game. There are six identifiable roles in Scrum that have different tasks and purposes during the process and its practices: Scrum master, Product owner, Scrum team, Customer, User, and Management. Scrum does not require or provide any specific software development method/practices to be used. Instead, it requires certain management practices and tools in various phases of scrum to avoid the chaos caused by unpredictability and complexity. The Scrum life cycle is shown in Figure 16.4. Scrum duration may vary from two to four weeks. Scrum includes *product backlog* which is a document that lists all the characteristics of software in a prioritized order. Product backlog has the same life cycle as the product itself. When a certain feature is implemented, it is removed from the list. The product backlog items are prioritized and then transferred to the *sprint backlog* which works as a task list in one sprint. It contains items from the product backlog which are planned to be implemented in one sprint. One product characteristic is divided into smaller tasks. New tasks cannot be added into the spring backlog when the sprint has started. Scrum practices include product backlog list (PBL), effort estimation, sprint, sprint planning meeting, sprint backlog list (SBL), daily scrum meeting (DSM), test, code sprint review meeting (R1), retrospective (R2), and release (R3).

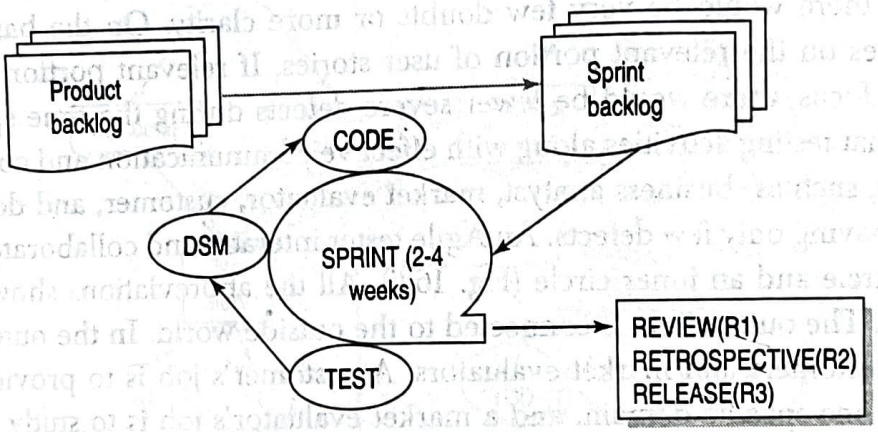

Figure 16.4 Scrum Life Cycle

16.5 Agile Testing

Since every team member in ASD intends to deliver a high-quality product, the agile tester does not wait for work but contributes throughout the development cycle. Contrary to traditional testing, unit test cases are written prior to the code writing, that is, these test cases are not in accordance with a formal requirement specification document. This is known as test driven development (TDD). Since Agile is iterative and incremental, the testers test each increment as soon as it is finished. Development and testing is done on a small feature so that we have a working code.

16.5.1 Test Driven Development

The idea of TDD is to write a production code (well-written code) along with testers which is passed through several rapid iterations. In TDD, each new feature begins with writing a test. This test must initially fail because it is written before the feature has been implemented. (If it does not fail, then either the proposed 'new' feature already exists or the test is defective.) To write a test, the developer must clearly understand the feature's specification and requirements. The developer can accomplish this through user stories that cover the requirements and exception conditions. This could also imply a variant or modification of an existing test. This is a differentiating feature of test-driven development

versus writing unit tests *after* the code is written: it makes the developer focus on the requirements *before* writing the code, a subtle but important difference.

These unit test cases are then run to ensure they fail. The next step is to write the code that will cause the test to pass. The new code written at this stage will not be perfect and may, for example, pass the test in an inelegant way. That is acceptable because later steps will improve and hone it.

If all the test cases now pass, the developer can be confident that the code meets all the tested requirements.

16.6 Agile Testing Life Cycle

Agile testing life cycle is based on the 'more is less' principle which focuses on communication management among stakeholders. The adoption of this principle results in quality software product as shown in Figure 16.5. The foremost component of this principle is 'more interactions' among all stakeholders. If communication between the tester and other stakeholders is very frequent and effective then there would be very few doubts or more clarity. On the basis of more clarity, team focus increases on the relevant portion of user stories. If relevant portion of user stories are covered with high focus, there would be fewer severe defects during the time span of sprint. This principle ensures that testing activities along with effective communication and collaboration among major stakeholders, such as business analyst, market evaluator, customer, and developer, results in software products having only few defects. An Agile tester interacts and collaborates with two circles, namely an outer circle and an inner circle (Fig. 16.6). All the abbreviations shown in Fig. 16.6 are show in Table 16.2. The outer circle is connected to the outside world. In the outer circle, the tester collaborates with customers and market evaluators. A customer's job is to provide an informal set of requirements in one specific domain, and a market evaluator's job is to study the market trends of the same domain. Further, from the outer circle, a tester may extract the latest technology trend, competitor's software product features, and the latest market standard. All these factors are analysed by the market evaluator and an updated set of data is provided during the user story finalization meeting in the presence of the Product Owner (PO). The PO's role is to satisfy the customer in terms of available bandwidth and expertise of the team while converting the informal requirements into a formal set of requirements known as the user story.

After finalizing the user story, the tester converts that user story into a ready story. This ready story acts like a checklist at the time of verification or acceptance of the user story by the customer. This ready story is the outcome of performing two types of testing. The types of testing carried out for a specific user story are:

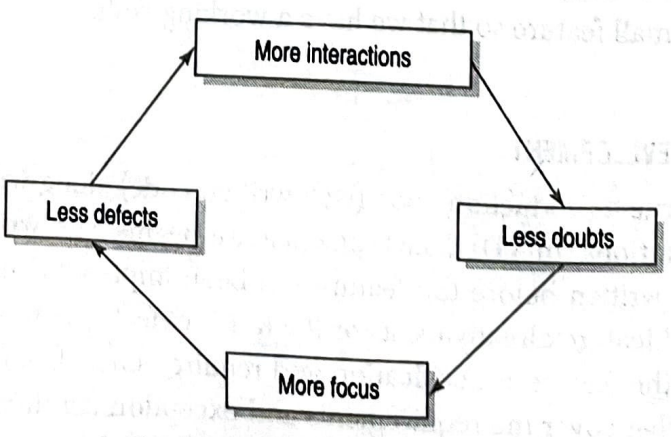

Figure 16.5 'More is less' principle

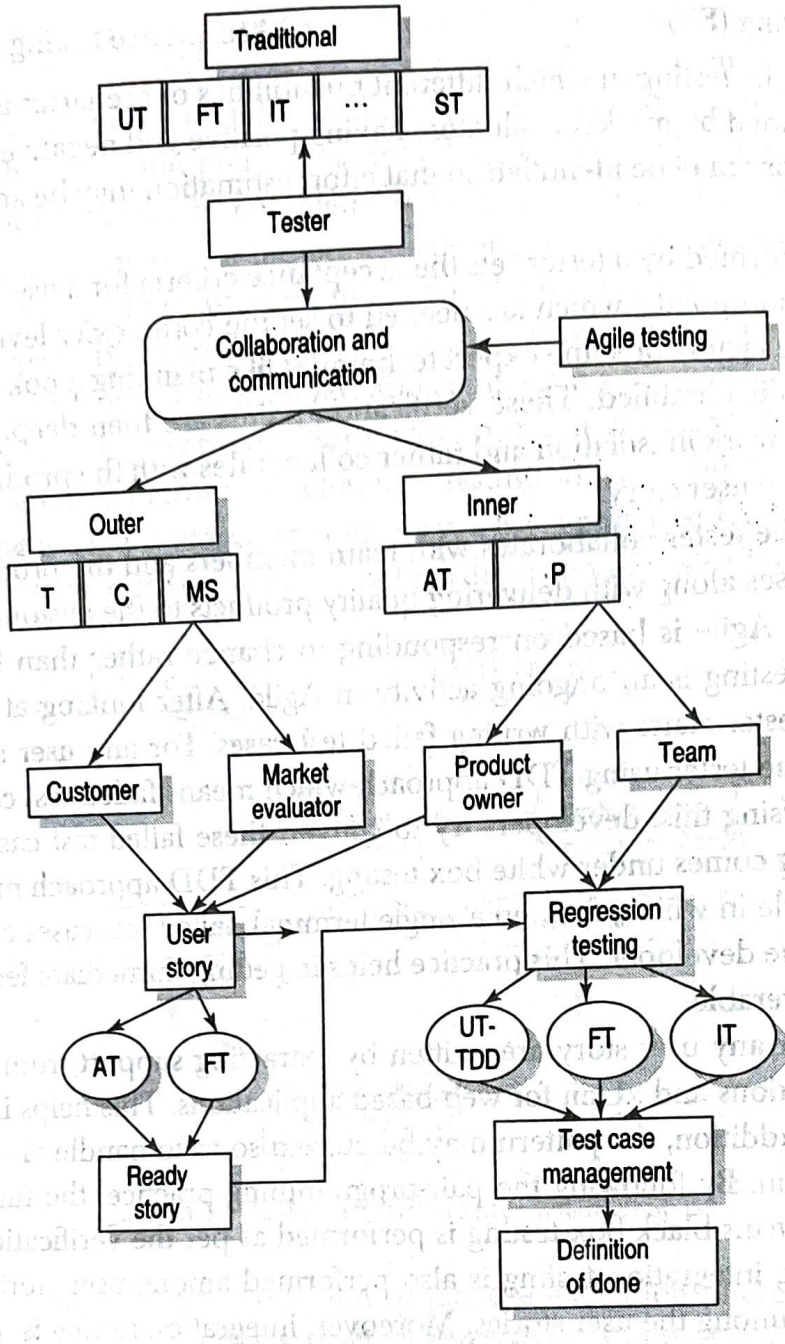

Figure 16.6 Agile testing life cycle

Table 16.2 Agile testing abbreviations

Abbreviation	Full Form
UT	Unit testing
FT	Functional testing
IT	Integration testing
ST	System testing
T	Technology
C	Competitor

Abbreviation	Full Form
MS	Market standard
AT	Automated tool
P	Pattern
AT	Acceptance testing
ET	Exploratory testing
TDD	Test driven development

- Exploratory testing (ET)
- Acceptance testing (AT)

Exploratory testing is the testing in which different possibilities or scenarios are considered as per the market analysis performed by market evaluators having positive and negative limitations. At the same time, the risk of user story may be identified so that effort estimation may be accurate in terms of effort, complexity, and time.

Acceptance testing performed by a tester sets the acceptance criteria for a user story. These criteria are also known as verification points, which are needed to set the complexity level of the user story. Specifically, based on some factors or some expert techniques like planning a poker game, the complexity level of the user stories is identified. These verification points are then deeply analysed. In this case also, the tester does not work in isolation and rather collaborates with the product owner to finalize the acceptance criteria of the user story.

In the inner circle, the tester collaborates with team members and the product owner. The tester's job is to manage test cases along with delivering quality products to the customer. In Agile, regression testing is important, as Agile is based on responding to change rather than following a fixed plan. Therefore, regression testing is an ongoing activity in Agile. After looking at the verification points of the user story, the tester starts with writing failed test cases. For any user story of the sprint, test cases are designed by the tester using TDD approach which means failed test cases are written for the upcoming user story. Using this, developers try to convert these failed test cases into pass test cases. This approach of testing comes under white box testing. This TDD approach may be implemented in a pair-programming style in which, first, on a single terminal, failed test cases are written, after which the code is written by the developer. This practice helps in getting immediate feedback so as to embed quality in the final deliverable.

Further, unit tests for any user story are written by extracting support from automated tools like Eclipse for Java applications and xUnit for web-based applications. This helps in reducing the overall time for any sprint. In addition, the pattern may be utilized so as to handle an existing problem with the best evolved solution. By following the pair-programming practice, the functionality of the user story is checked; this means black box testing is performed as per the verification points of the ready story. During the sprint, integration testing is also performed among user stories of a sprint by considering dependencies among the user stories. Moreover, integration testing is also performed among user stories of the different sprints. Further, to manage test cases, effective regression techniques, such as regression test selection (RTS) and test case prioritization (TCP), are implemented so as to run only a subset of the test cases out of all the test cases.

Finally, depending upon the feedback cycle of customers, the product is released by the operational team in collaboration with the tester by performing all necessary testing including usability, scalability, etc. Feedback of customer is an important input for getting good quality product. Finally, 'definition of done' is declared by the customer after matching the verification points of the ready story with the actual product. This is an easy way to check the validity of user stories in a sprint.

16.7 Testing in Scrum Phases

Scrum methodology is based upon small duration sprints having a small number of user stories listed in the sprint backlog list (SBL). SBL is a subset of PBL. After carrying out effort and complexity estimation by PO, SBL is finalized. This methodology is divided into three phases. The Scrum phases are pre-execution phase, execution phase, and post-execution phase. In this section, all the testing activities

occurring before, within, and after the sprint have been identified. For the purpose of simplicity, only three sprints, namely, S1, S2, and S3, are taken in the sprint flow diagram having three phases (see Figures 16.7–16.9). The duration of execution of these sprints is W1, W2, and W3, respectively, where W stands for week. In the execution phase, a sprint S1 may be completed having n number of user stories from SBL1. Similarly, S2 may be completed having m number of user stories from SBL2.

Figure 16.7 shows testing scenarios in the pre-execution phase of Scrum methodology. This phase starts with collaboration among the customer, market evaluator, product owner, and tester. They sit together to finalize the user story and ready story. The ready story is based upon the confirming points which are as per the market standard, technology, and competitor's product features. These confirming points are also known as acceptance criteria. During the sprint, these acceptance criteria are frequently checked. In addition, the tester performs exploratory testing so as to check the feasibility of various scenarios. After finalizing the ready story and user story, a list is prepared having all the finalized set of user stories. This list is known as PBL which is input for the second phase, that is, the execution phase.

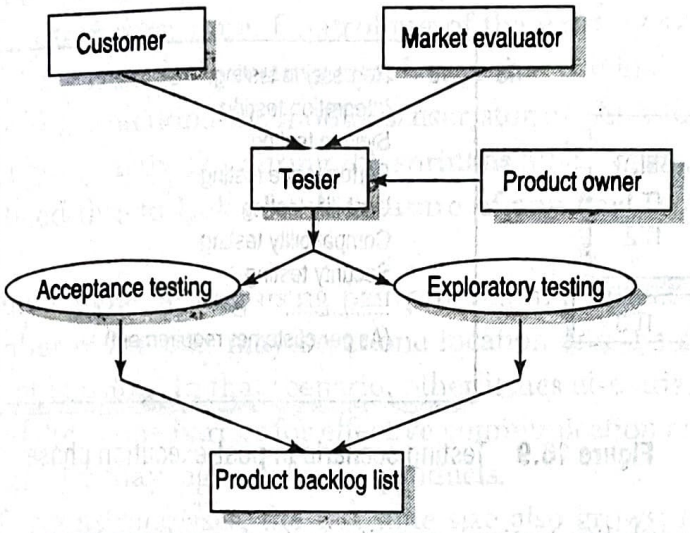

Figure 16.7 Testing scenario in pre-execution phase

After receiving input from the pre-execution phase, the execution phase starts, which is shown in Fig. 16.8. PBL is analysed by the PO and effort estimation is done for selecting the user stories for SBL1 and SBL2. SBL1 and SBL2 are executed in sprint S1 and S2, respectively. In S1, the tester performs unit testing with TDD or white box testing, functional testing or black-box testing, regression testing, integration testing among dependent user stories, and many more depending on the requirements set by the customer. The output of S1 is an integrated set of user stories, IT1 with regression test suite during W1 duration. Similarly, in sprint S2, the same types of testing are performed–an integrated set of user stories IT2 with regression test suite during W2 duration. Further, these integrated sets of user stories IT1 and IT2 are considered user stories in SBL3. Further, there may be other user stories which need to be developed in W3 duration which are newly added features in the maintenance time of the product. SBL3 is input for the post-execution phase which is shown in Figure 16.9.

In post execution phase, user stories are selected from SBL3, based on the priority set by the customer, complexity level, risk level, or any other prioritization factor. In this phase, different types of testing are performed based on the customer need. Various types of testing that are mandatory in S3 are integration of IT1 and IT2, functional testing, system testing, and regression testing depending on the modification suggested by the customer, if any. Other optional testing that may be performed

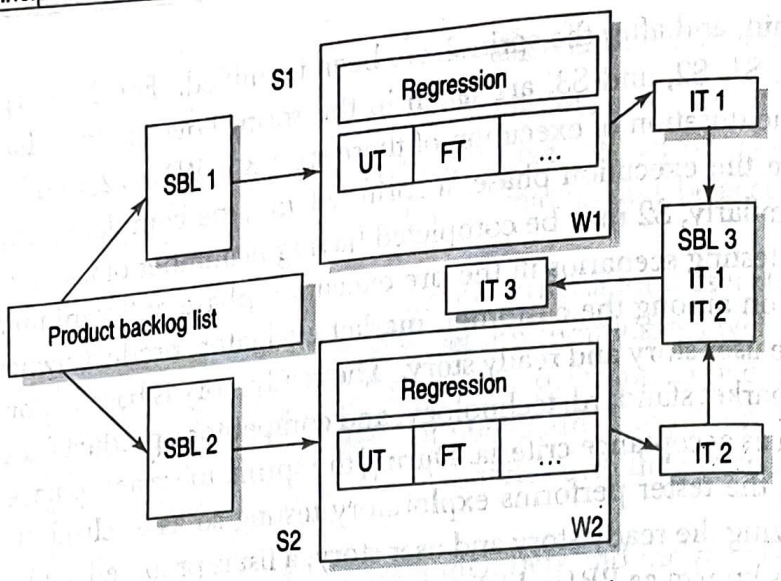

Figure 16.8 Testing scenario in execution phase

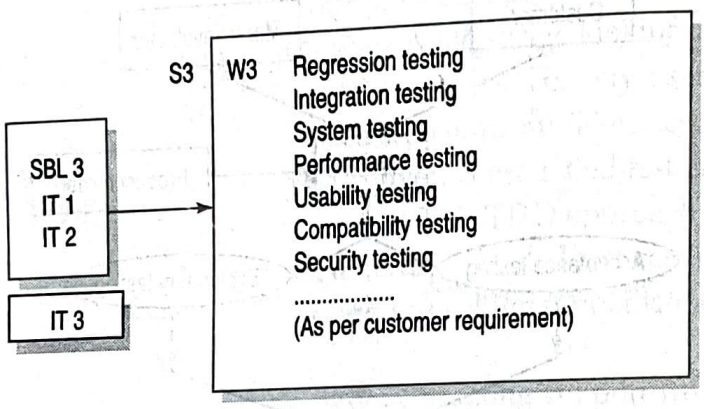

Figure 16.9 Testing scenario in post-execution phase

in W3 duration are compatibility testing, security testing, performance testing, usability testing, etc. Finally, a software product is delivered to the customer.

16.7.1 Regression Testing in Agile

Regression testing in Agile environment is practised under two major categories.

- Sprint level regression testing (SLRT): It is focused on testing new functionalities that have been incorporated since the last production release.
- End-to-end regression testing (EERT): This refers to the regression testing that incorporates all the fundamental functionalities.

Each sprint cycle is followed by a small span of SLRT. The completed code goes through further regression cycles but is not released into production. After few successful sprint cycles–typically three to four–the application goes through one round of EERT before being released to production.

16.8 Challenges Related To Agile Testing

In ASD, testing is not performed in a single phase like traditional models; rather it is an ongoing activity. Hence, Agile testing is a challenging task. The challenges to Agile testing are as follows:

- The basic principle of Agile methodology is responding to change than following a strict plan. The occurrence of frequent changes in Agile may have crucial after-effects if necessary steps are not taken at the right time. Thus, a cumulated test suite may become a hurdle after a number of sprints in a large-scale project. So, there is a need to perform regression testing using effective techniques so as to reduce the size of pending backlogs of user stories and the test suite respectively.
- In ASD, one of the representatives of the customer is always present at the development site to give instant feedback and for any future improvement in the sprint. Thus, work to be delivered to the customer is frequent and response is also frequent from the customer side. This response may comprise improvement in the existing system, new requirement, new work style, scalability of the existing system, etc. At the same time, customers may furnish new requirements that may disturb the original functioning of the existing system. These changes that are introduced later may have several unnoticed effects in the working system. These effects must be controlled in a planned manner by team members so as to deliver the quality deliverable to the customer on time. Controlling of the existing system is the first priority as per the definition of the regression testing which says that original modules should not regress by introduction of new functionality/modules/user stories. Although unit testing and acceptance testing are ongoing activities during the sprint as a part of regression testing, some bugs may still go unnoticed due to lack of risk measure of any new requirement disclosed by the customer.
- Further, in distributed Agile, testing using pair-programming practice may be cumbersome as the first team member of the pair may be at one location and the second member of the pair may be at a different location. In that scenario, other issues also arise, such as language barrier, cultural barrier, and time zone barrier for effective communication among team members of the sprint. Therefore, quality may lag in software products.
- As the number of sprints increases, the test suite size also grows; hence, management of test cases becomes a problem in a distributed environment.

SUMMARY

Let us quickly review the important concepts described in this chapter.

- Agile Software Development (ASD) is an iterative and incremental development method and its basic concept is customer-centred and it acknowledges that requirements can change.
- The key feature of ASD is effective communication between all team members. For valuable communication and information exchange, daily meetings are held in ASD.
- Agile methods break tasks into small increments with minimal planning, and do not directly involve long-term planning.
- Scrum is a lightweight method to iteratively deliver software in which changes are always welcome.
- The Scrum process includes three phases: Pre-game, Development, and Post-game.
- There are six identifiable roles in Scrum that have different tasks and purposes during the process and its practices: Scrum master, Product owner, Scrum team, Customer, User, and Management.
- Since Agile is an iterative and incremental model, testers test each increment as soon as it is finished. The development and testing is done on a small feature so that we have a working code.