CHAPTER 6

ACTIVITY OF TEAM MEMBERS

Chapter 5 described about the different roles shouldered by agile coach. This chapter discusses the responsibilities of the individual team members, as well as the shift in emphasis in their work during transitioning from traditional software development to agile software development approaches. For the new comers agile practices will be listed in the scrum board before start of sprint. This was pointed out by an experienced developer as, “My scrum master will chart the list of practices that are available in scrum framework. He does this because agile approach in general is not prescriptive and we as team members feel comfortable in handling the task” — Prac 57.

Shift in emphasis includes activities that work on a day to day basis. Such activities are explained in the following sections in a detailed fashion.

6.1 SELF ORGANIZATION VERSUS PRE-DEFINED ORGANIZATION

One of the main practices in agile software development is self organization. Self Organization requires each member to choose or justify his or her duties in a team that is natural for all living beings.

An experience was shared by an agile global strategist as, “Self Organization is available in the origin of human life itself; it is the self maintaining nature of ‘biological’ systems from a cell to the whole organism, pattern formation and to identify how the living organism develops and grows. We observe creation of structures by the behavior of social animals such as social insects (bees, ants, termites), many mammals and the formation of flocks by birds, fish etc” — Prac 47. Such behavior expected of agile teams to be noted here.

6.1.1 Real Time Project Training Over Classroom Training

Traditional class room training is effective for transferring explicit knowledge that is well established and clearly (explicitly or unambiguously) documented. Such general knowledge with varying degrees of details is often insufficient to perform a real job at hand. When a trainee observes or partners an experienced practitioner doing the above job, he gets to know all the necessary details of the procedures which enables him perform similar jobs independently. This part of the knowledge, often difficult for explicit documentation is called the tacit knowledge. Agile team members become productive faster through such hands on training. This tacit knowledge is valid only for one particular technology currently used unlike the explicit knowledge with more general validity provided through class room lecture. But it meets the immediate requirements of job performance and provides opportunities for getting aligned with the approach adopted by experienced agile developers.

6.1.2 Required Documentation Over Entire Documentation

The agile software development project supports required documentation. Some practitioners have a wrong notion and claim that agile software development does not require documentation. Documentation is necessary even when working with agile projects. But, only documents that are vital to the task are created. Importance to the complete end to end documentation is not emphasized in agile projects. This was pointed out by a developer as, "Only vital information that are really important to the working of the software are documented" — Prac 20. Such documents created will be kept in the document repository of the project. The document copy should also be circulated to customer for his perusal for feedback and for his future reference.

6.1.3 Liberty and Responsibility

Traditional software projects will have their own hierarchy as team lead, project lead, project manager and so on. Project manager of traditional projects will assign task, will set deadlines and will monitor the work on a daily basis. (Prac 10, Prac 20, Prac 36, Prac 53). Agile team members have the liberty at their work and at the same time they know their commitment and their responsibility. Members who are willing to transition from traditional to agile development will feel the difference in their work. They feel independent and they are at liberty to identify task and to execute it.

Agile practices makes team members self motivate, self assign the task and to self organize (Prac 1-Prac 7, Prac 8-Prac 13, Prac 20-Prac 25, Prac 26-29, Prac 30-33, Prac 53-Prac 56, Prac 61-65, Prac 66 - 68), thereby getting a feel of self empowerment. Self empowered teams become sole owner of fulfilling the task assigned by them rather than getting assigned by manager as carried out in traditional development approach. Team members pick their task from story boards or scrum boards, and individual team members evaluate their stand through daily stand up meetings and team members evaluate themselves through retrospective meetings. Project progress is monitored with the help of information radiators such as burn down charts and story boards.

6.1.3.1 Team effort in planning, estimating and decision making

Product owner or the customer representative provides requirements in the form of user stories. User stories are divided into small tasks by team members during the planning meetings. Based on small tasks, the number of iterations is estimated. Scrum master allows the team to work on estimation. Story points are decided based on collective decision making. Story points are the number of burn hours required to complete the specified task. During the initial days after transition from traditional development approach, the team may not be comfortable in estimating and planning and situation changes as experience level in working with agile projects pile up. Contrast this with the traditional approaches where project manager will take care of estimation and planning as told to them leaving team members to make them work.

A developer has shared his experience as, “That was the first meeting I had after transitioning from traditional development to the agile development. The team members gave me few cards and I did not know what to do with the cards. Then the coach taught me how to work with estimation and now after a few months I could size something without the cards and knew exactly what may be the size of a story without even thinking about the card. It was so natural that I got used to it” — Prac 2.

6.1.3.2 Self assessment

Scrum team and individual member progress are assessed through daily scrum meeting. A scrum meeting will last long for a maximum duration of fifteen minutes and only three vital questions are asked. They are listed as,

“What was done the previous day? ,
What is proposed to be done?
What are the issues identified?”

The daily standup meeting will act as a good practice that will give shared responsibility among team members. Product owner, scrum master and team members are present in the meeting.

6.1.4 Observation of a Daily Standup Meeting Conducted in an Organization

Typical meeting proceedings are as, Teams were ready for their daily standup — Four developers and one Scrum Master. The meeting was crisp and concise. The members discussed what they had achieved the previous day and then discussed what they plan for the current day. They were also able to spot and resolve dependencies and impediments by informing each other of their daily progress. There was a relaxed and professional behavior throughout the meeting venue. Other than this, no other questions were raised. The meeting lasted for about twelve minutes. The Scrum Master supplied additional information for resolving certain issues and asked one of the team members to resolve the issue that another team member faced. Then the team went back to their regular work.

The daily standup meeting will serve as a self-checking tool. Daily progress of the Team members is known to each other. Any slackness in progress will be immediately visible during a meeting and brings on peer-pressure to deliver.

6.1.4.1 Use of information radiators

An information radiator radiates the project progress information with high visibility. Agile teams keep track of the progress using their own metrics. Some of the common metrics used in agile development projects are size, effort, velocity, burn down, time to market and the cycle time. Story and story points available in story boards, task, feature occupy the size metrics.

Actual hours taken along with days, weeks, months, years occupies the effort metrics, features and iteration occupy the velocity. Benefit of using metrics can check project progress, productivity and predictability of tasks. The areas of improvement can be identified; thereby the performance and motivation of team members can be improved.

Table 6.1 Terminology and its description

Terminology and Description	**Used by**
Velocity is a measure of how much of user story points get completed in a given amount of time. Only the completed story points count towards velocity.	Individual and team
Burn down chart is a graph that traces the number of complexity points remaining versus the number of iterations, indicate the burn down rate. Usually, the burn down chart will feature the ideal burn down rate needed to achieve the iteration goal.	
Cycle time is the number of days or the iterations taken to complete a story. It is also called as the average time between the deliveries of the completed work items.	

6.1.4.2 Story boards

Story boards and sticky notes are prevalent when working with scrum framework. Members take up a story, write them on sticky notes and fix them onto story board or scrum board. Story board will be classified in to “To Do”, “Doing” and “Done”. “To Do” category tasks are yet to be done. Tasks will be written in small white paper and will be stuck onto scrum board. Team member can pick up task that he / she desires. Business priority assigned to the task is marked on the note and the member is expected to pick the task based on the business priority and not fully based on his / her own interest. An experience shared by one of the developer is as, “The intention is to deliver business value as soon as possible - that you take items that are most vital from point of view of business” — Prac 57.

Team member has to write his name in the sticky note and have to place in the “Doing” category. When task is completed by a particular team member it has to be moved to the “Done” category. This makes a team member to self evaluate his strength and to self assign his task and gets self motivated by working out the task that member desires. Figure 6.1 show the sticky notes fixed on the story board as found in the office of an Indian Software Organization.

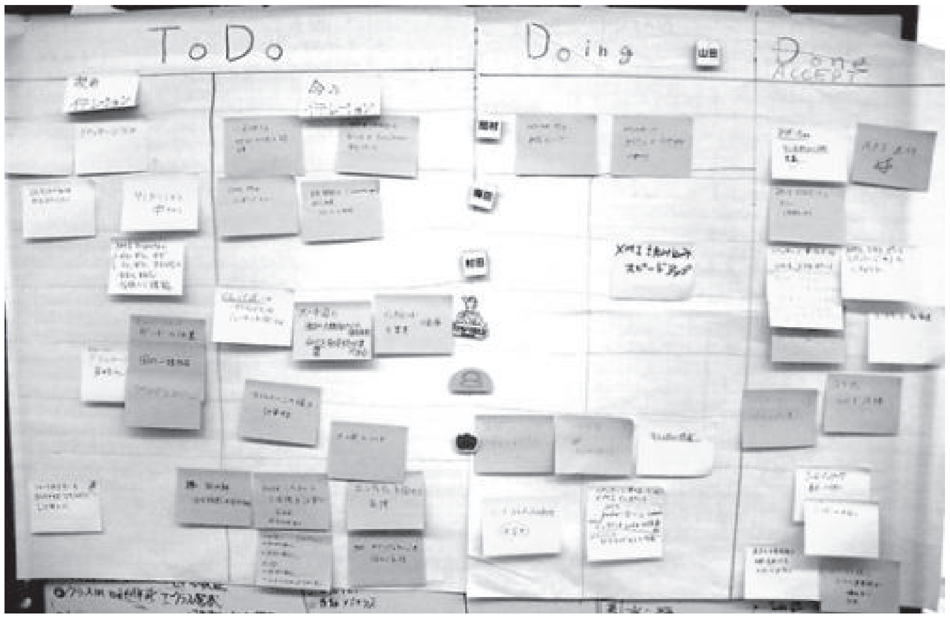


Figure 6.1 Sticky notes on a scrum / story board
Source: Photograph taken from a software organization

The stage “Done” can be achieved only when “To Do” items is completed in all respect and if it is checked in to the source control and has to get a sign off from the client thereby remaining hours of task becomes zero.

6.1.4.3 Burn down chart

Burn down charts is classified into two groups, viz., the release burn down chart and the sprint burn down chart. Project progress is measured using a release burn down chart.

Release burn down chart

Release burn down chart is a graph which helps visualize total points remaining for release at the end of each sprint. A typical release burn down chart is shown in Figure 6.2.

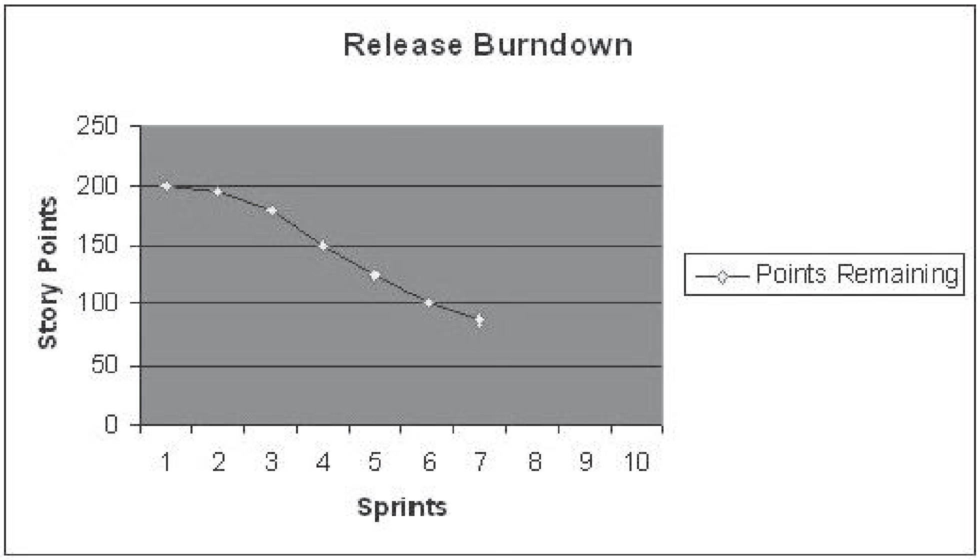


Figure 6.2 Release Burn down chart (Chandrasekaran 2012)

Sprint Burn down chart

Sprint burn down chart shows estimated number of hours required to complete sprint tasks. A sprint holds the number of iterations and minimum marketable feature delivery shown to the customer at the end of every sprint. Usually, each sprint holds two iterations. Sprint duration varies from a fortnight to a month. With the help of sprint burn down chart, customer can have a visual representation of functionality delivered, and the functionality that are yet to be delivered and overall team productivity. Figure 6.3 illustrates the sprint burn down chart. This was pointed out by one of the developer as, “Sprint burn down makes the customer at ease. They can know the tasks that are to be delivered to them. They can have a rough idea of how things are progressing” —— Prac 61.

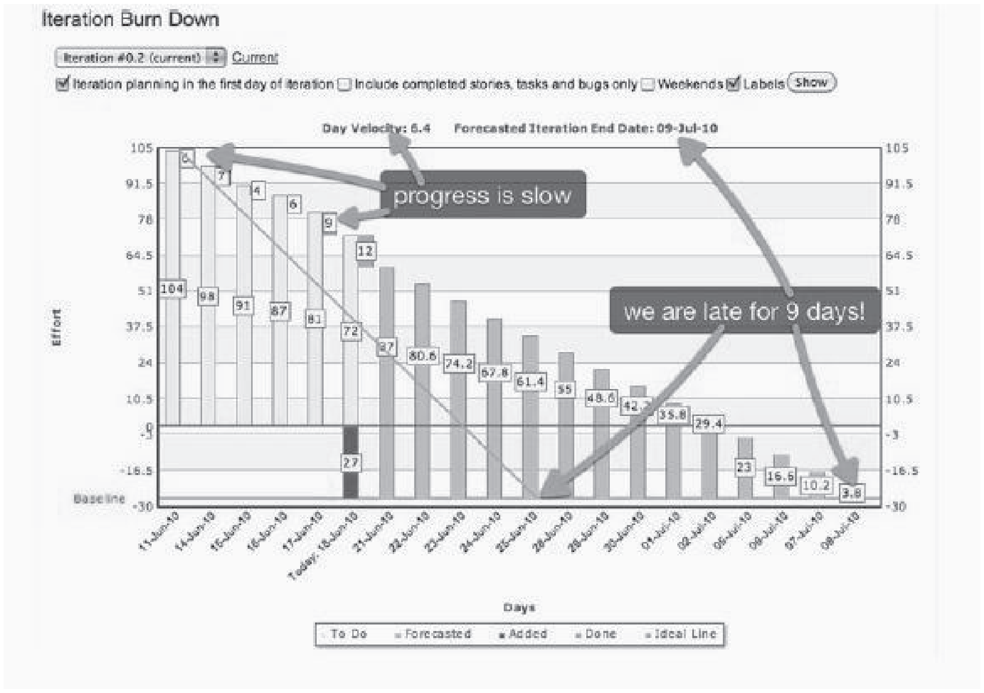


Figure 6.3 Sprint burn down chart (Chandrasekaran 2012)

6.1.5 Issues Identified in Liberty and Responsibility

There are chances that the experienced developers working in agile software projects may try to overpower the new comers in the team and will try to misuse the liberty given to the senior members. It is the responsibility of the agile coach to identify such practitioners and should correct them through a discussion with the team and with the support of the top management.

6.2 SHIFT IN EMPHASIS BETWEEN GENERALIST AND SPECIALIST

In agile projects, team members do not narrow down to only a very specific role on projects. They act on different roles thereby team gets people with good amount of exposure on all the roles of project. Although team members are allowed to work on multiple roles, they need to be a specialist in a particular technical and functional area.

In traditional development approach, the team used a linear approach, wherein developers work with design and coding; the testers work with testing and so on till the end of the project. Whereas in agile software development projects, the developer involved in coding and the testers together work on generating several test cases and executing them. Above scenario prevails in newly transitioned agile development teams. This scenario is depicted in Figure 6.4. As team gets considerable amount of exposure towards agile development, they interchange their roles at the start of new iteration. This way they become generalists in playing multiple roles that could facilitate rotation during the tenure of the project. Figure 6.5 illustrates this scenario. By rotation, team members get a broad exposure exposing themselves to new concepts and their technical area of expertise gets enhanced.

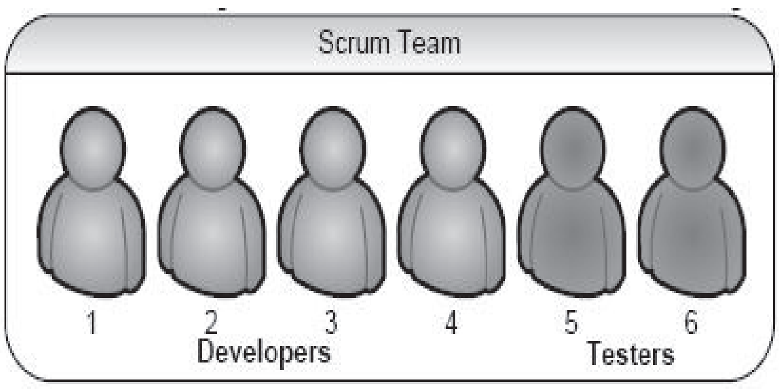


Figure 6.4 The scrum team

Advantage of being a generalist in team makes team members learn many new things. In agile projects that deploy a combination of scrum and XP methodology, a developer whoever is involved in coding can be converted to play testing role. Normally, a sprint lasts for a period between fifteen days to one month. In this case, as shown in Figure 6.2, both tester and developer will be happy in playing multiple roles. As developer would have played the role of a tester in the same project, he would be doubly careful in coding the task that is assigned to him avoiding the repetition of defects unearthed during earlier testing.

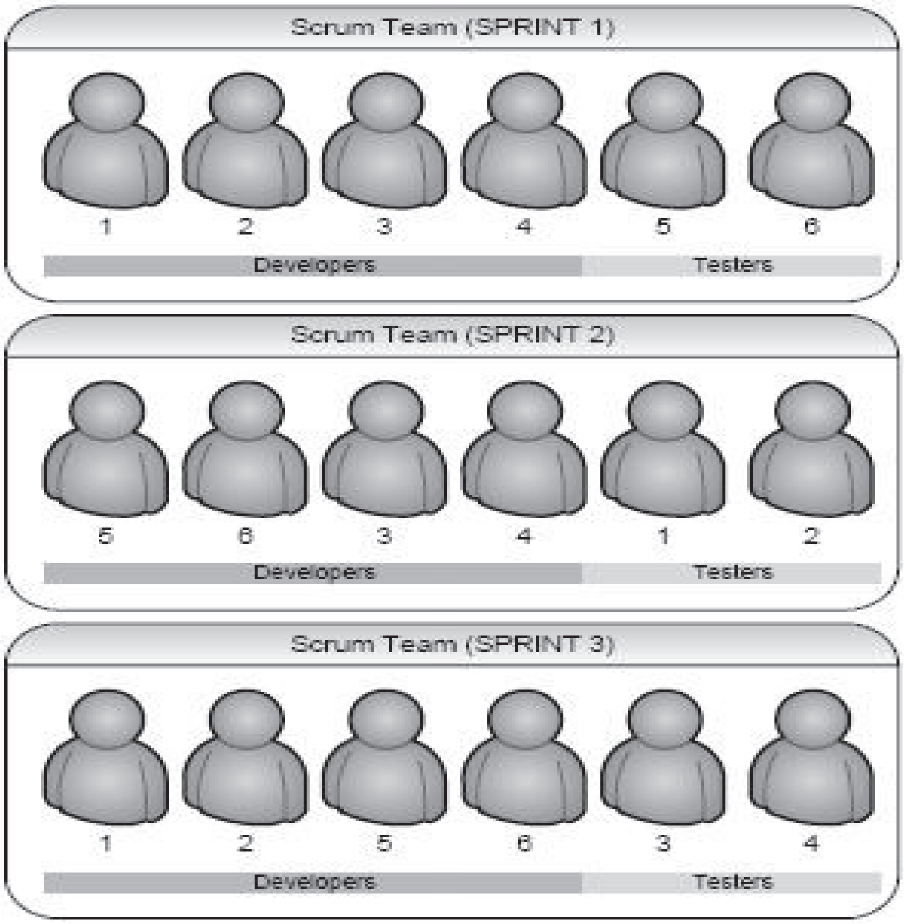


Figure 6.5 Rotation of roles played by the team members on the scrum team

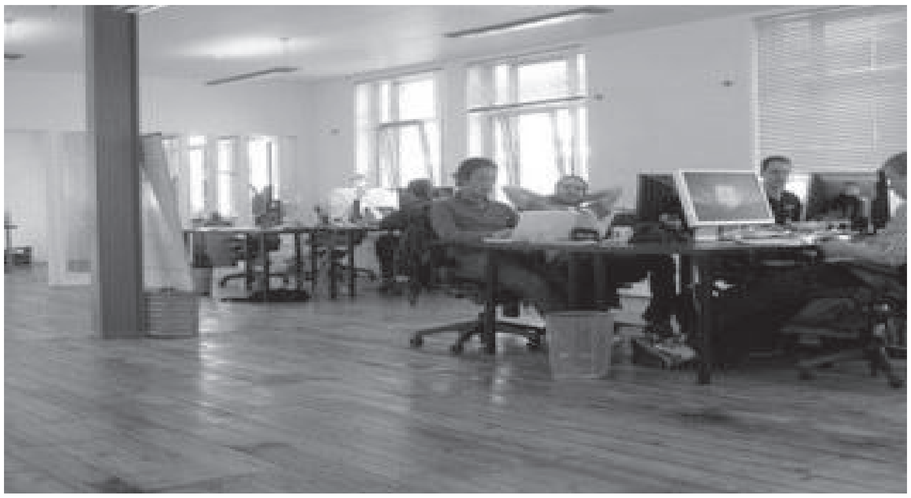
One of the business analysts pertaining to agile software development team has shared his experiences, "During the iteration planning meeting, the scrum master asks us to choose a role. Initially, we choose the role that we are familiar with. As the project progresses, the business analyst does not have much role to play, so he will opt for a different role"——Prac 39.

Figure 6.5 indicates that the developer changes his role from coding to testing and the tester changes his role from testing to coding. The scrum master or the agile coach should take extreme care about the team members when their roles are changed during the initial days of their work. Testing of agile projects is the area where the developers of the agile teams will have a passion to do it. But, when a developer wishes to change the role with the consent of the scrum master, they need to follow certain recommendations on the Do's and Don'ts of testing. A questionnaire to determine whether a developer will be in a position to shift himself to testing and the recommendations which will be highly helpful for an agile developer cum tester in an agile developmental project.

One of the scrum masters have claimed his view point on the above issue as, "Without the consent of the scrum master, the team members are not supposed to switch their roles. The major risk involved is that there are chances that they may ruin the code and waste the project iteration time. The team members have to be properly trained before they could opt for a new role" —— Prac 38.

There is always a chance of doubt when an individual takes up a task outside the area of his expertise.

Group programming will improve data gathering and analysis and will improve the code base as two or more members stretch their helping hand in giving ideas for a task. The team members should work on an open space environment with no cubicles. This will result in improved communication and the learning curve of the team members. The main intention of group programming is that it puts developers and testers together in the same physical space. As a result the developers and testers work together on one team. Developers value the testers' perspectives and vice versa. Figure 6.6 illustrates this scenario.



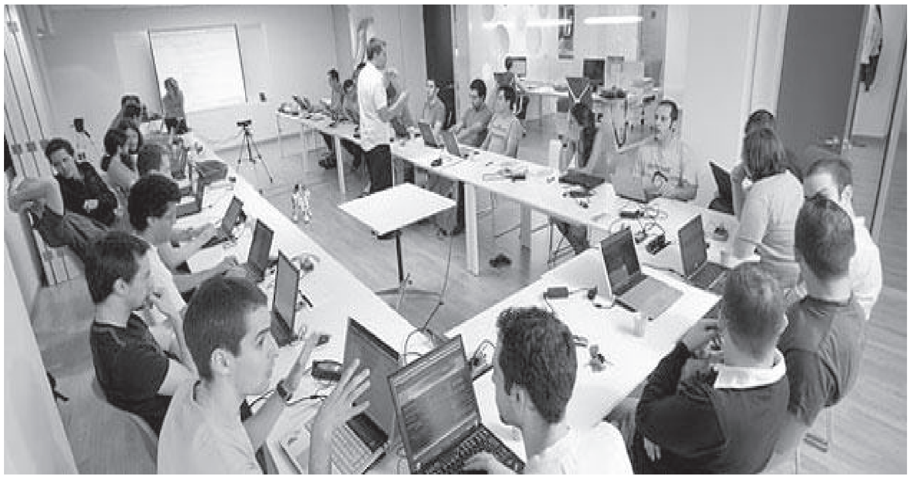


Figure 6.6 Agile open work space environment
Source: www.blog.new-bamboo.co.uk/2009/8/6/knowledge-sharing-i-open-workspace-and-code-reviews

6.3 INCESSANT LEARNING OVER RETROSPECTIVES

6.3.1 Incessant Learning

Incessant learning or continuous learning is a strong requirement all over the software industry. The learning curve and the ability to grasp the contents differ from person to person. Incessant learning in agile practices involves different types of learning. The team members have to learn the set of agile practices, learn new skills based on technology development and become a specialist.

6.3.2 Retrospective Meeting

Best way to learn things is by learning through experience. Retrospective meeting is one of the best places where the team members can learn from the faults or good experiences that they have encountered in their projects. It identifies and implements improvement actions. After every sprint gets completed, most of the organizations follow a break day. A break day is a day where the team will be checking number of backlogs still pending in the product backlog. A retrospective meeting is conducted in order to check the tasks that went on well, the tasks that had problems, the tasks that were dumped in the product backlog and the common problems faced by the team during the sprint cycle.

A developer has shared his experience as, “We, the team will have a tensed situation at the end of every sprint. After delivering the minimum marketable feature to the client, the members of the team will be highly relieved. This is similar to time of seeing an examination result” —— Prac 66.

Recommendations for the conduct of retrospective meet including tasks that are to be carried out before the meeting, during the meeting and after the meeting are listed in the Appendix 7 of this thesis.

Retrospective meetings provide valuable opportunities for the team to learn from their past experience. After the meet they are in a position to evaluate them and improve their future potential (Derby and Larsen 2006).

One of the scrum masters has shared his experience as, “Learning new concepts need not be only from peers. The team members are allowed to attend seminars, work shops, brain storming sessions and the like whenever they are free. Our team members share their knowledge also during lunch sessions (called the “Brown Bag lunch sessions”), thereby making an effective utilization of time” —– Prac 38.

Retrospective meetings also help assess iteration pressure and come out with suggestions to overcome or at least minimize.

An experience was shared by one of the testers as, “Retrospectives are the key ingredients in agile methodology which make the team evaluate team practices and fetch remedy in correcting them” —– Prac 25.

6.4 REQUIRED LEARNING OVER COMPLETE LEARNING

Pressure during iteration period is a common factor for the members of the team. With iteration pressure, it would be very difficult for team members to make a systematic end to end learning on a subject matter. So, the team members will learn only the needed concepts and procedures relevant to the project context. This is in sharp contrast with the learning style of traditional software developers.

6.5 CO-LOCATED TEAMS OVER GEOGRAPHICALLY DISTRIBUTED TEAMS

Co-located teams are located within a viable distance that enables frequent face to face meet of members. This in turn reduces communication gap and project issues are resolved with immediate effect.

In a geographically distributed team, team members may be spread across entire stretch of the globe. Physical meeting of such team members on a day to day basis is not possible. Meetings among such team members are conducted through video conferencing and chats. In such a scenario, there are large amount of chances to get involved in communication gaps amplified by any cultural differences.

When customers are on a remote location, gathering requirements from them and explaining the customer about the marketable deliverables become a big challenge. Some of the issues identified in gathering requirements are illustrated below as a case study.

6.5.1 A Case Study

6.5.1.1 Distributing activities globally

Reasons for offshore software development in agile projects

The main reasons for company to move offshore are to reduce cost of development due to lower wages, skilled labor and round the clock development. The reasons for moving offshore are given below:

i. Improvement in Telecommunications

ii. Favourable Government policies

iii. Skilled and very cheap man power

iv. Less cost of setting offshore site

Detail requirements and design specification are sent to offshore site where they can develop software. This approach is useful to minimize the communication in delivering the product that is taken up on an offshore development. This will create more confidence particularly in offshore team to allow decision making in the team. Good amount of communication takes place between onshore and offshore teams resulting in much better job performance. The Agile project puts great focus on communication with the end-user or customer. Danait (2005) claims that communication can help in developing trust amongst both the onshore and offshore teams. The major challenge in understanding the software requirements lies through effective communication and knowledge sharing, domain knowledge, less software requirements documentation.

6.5.1.2 Issues identified in gathering requirements

The case study was framed based on the researcher's involvement in an agile software project that has been developed in a software company located in Chennai, India. The same product had to be served to three different customers located in remote locations like China, Germany and Philippines. Interaction and minimum marketable feature demos are shown to the customers through video conferencing and chat engines. Company handles projects on image storage optimization using image compression. They are also leaders in providing solutions for correcting Optical Markup Reader (OMR) sheets for online competitive examinations. The company has its offices in various locations across India along with global operations in USA, Japan and Philippines. As per the policies of the Organization, the company name and the Project names are kept confidential. A beta version of the product to compress the image and retrieve it back without any visible loss of information has been released.

6.5.1.2.1 The Analysis Team

Figure 6.7 shows the members in the analysis team.

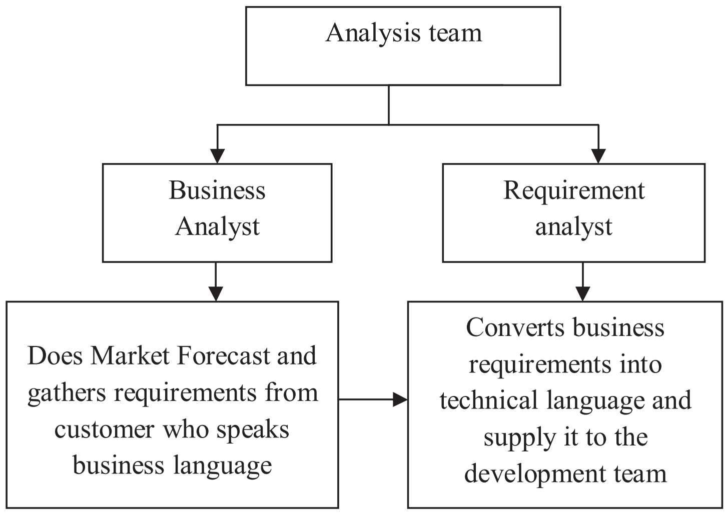

Analysis team
Business Analyst
Requirement analyst
Does Market Forecast and gathers requirements from customer who speaks business language
Converts business requirements into technical language and supply it to the development team

Figure 6.7 Analysis team

In this Organization, delivery head acts as requirement analyst. He allocates resources and schedules tasks for developers and also responsible for fixing deadlines for frequent deliveries.

The code written was integrated three to four times a day and a minimum marketable feature was generated at the end of each day. This feature was shown to the customer representative and feedback from them was obtained for further improvements. Continuous integration with continuous delivery and continuous feedback from the customer end and its follow up actions improve the relationship between the customer and the team and thereby it improves quality of product delivered.

6.5.1.2.2 The Project Team

The illustration in Figure 6.8 shows the cross-functional team that includes a product manager, developmental team, Quality Assurance team, User Experience team, and a project manager acting as scrum master.

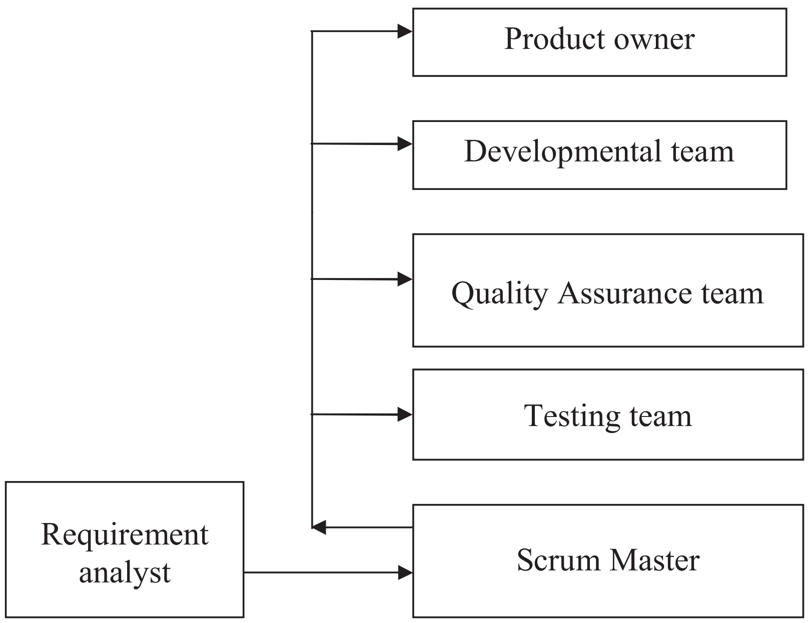


Figure 6.8 Project team

Product owner supervises release schedule and helps resolve logistical issues, but otherwise does not participate in the development process. Developmental team, consisting of three to eight developers, implements and delivers a feature or set of features per sprint. Quality Assurance (QA) team tests the features in that Sprint and identifies issues for further resolution. Development team fixes these issues in the current Sprint. The sprints that they generally adopt are two weeks in length, with seven of the ten working days of the Sprint dedicated to implementation by engineering and the remaining three days to Quality Assurance for testing.

The cross-functional development team for the project consisted of the product manager, five developers responsible for design and implementation, two QA members and two members of the testing team. The testing team was responsible for testing the intermediate delivery and will try to give suggestions on improvement of the product. All members of the team were located in a single office at Chennai, India.

6.5.1.2.3 Usage of Sprint on the Project

Development team was invited to view user testing sessions, discuss findings and review preliminary designs during the sprint session. This helped team to share their ideas, and enabled the team to prepare for next sprint. At the end of each sprint, final designs and specifications were reviewed and follow up action were made. Few tasks were placed in the product backlog to be taken up in the forthcoming sprints. Retrospective meets became the convenient forum to share the results of user testing, and to clarify any misinterpretations of designs.

6.5.1.2.4 Sprint Zero

Sprint zero takes place at the start of project. Figure 6.9 shows the task of sprint zero.

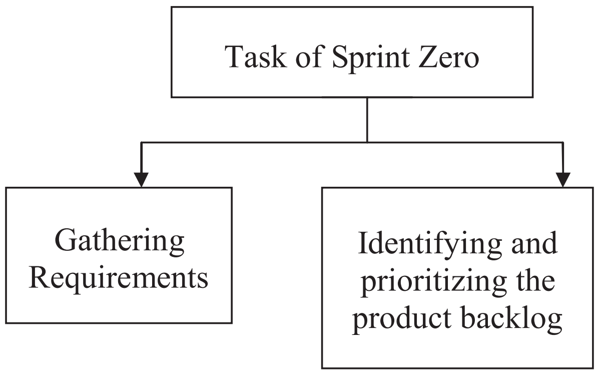


Figure 6.9 Sprint zero

Sprint zero is used by the team to review requirements and create initial user stories based on backlog items. Knowledge transfer of requirements were made to the team which utilized sprint zero to better understand users' needs, to explore their context and identify their goals for the project as a whole. Data obtained from the initial user research effort is used to negotiate the priorities of the first sprint and communicate the users' expectations to the cross functional team. User scenarios and the features were developed in subsequent Sprints. The customer response and follow up action is shown in Figure 6.10.

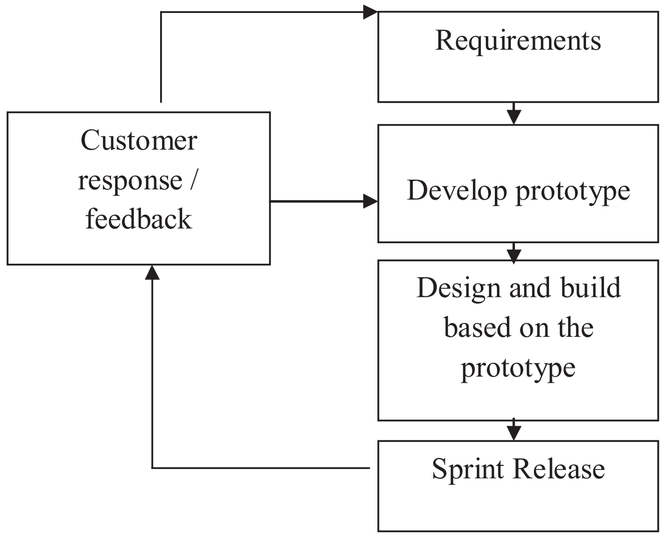

Requirements
Customer response / feedback
Develop prototype
Design and build based on the prototype
Sprint Release

Figure 6.10 Follow up action based on customer response

6.5.1.2.5 Challenges in Conducting Reviews

The challenge that the organization faced was in not getting the clients review early builds. Thus offshore team had to wait for a long time to get a response from the client. Validation of software requirements due to change in business priorities became a challenge. Lack of full business knowledge led to difficulties in software requirements validation by the offshore team. Requirement validation was carried out only through video conferencing with customer.

6.5.1.2.6 Conclusion

The challenges caused due to lack of communication, oral communication problem, cultural differences and difficulties in knowledge sharing were well realized. Based on some of the observations identified during this study, recommendations for co-located scrum team and globally distributed scrum team are cited in the Appendix 8 of this thesis.

6.6 RESEARCH FINDINGS AND THE RELATED LITERATURE

The research findings are organized based on distinct groups such as total alignment with the global practice and significant deviation from the global findings.

Dagenais et al (2010) states that the involvement of agile coach in the new team is beneficial for the team members to sync with the pace of the project at the earliest. Our research also claims that the involvement of agile coach helps the new comer settle down with the team at the earliest.

Cockburn and Highsmith (2001) have emphasized on people factors in terms of commitment, skill, talent and communication. Our research finding is that the practitioners communicate well by working in an open space environment and improve themselves by getting feedback from the customer on a regular basis. The agile coach is empowered to eliminate a team member, with the support of the top management, if he/she is not fit to work in an agile development team.

Agile software development approach does not work when the team members are kept in isolation from the team. They require support from the top management and good rapport with the customer. Moe et al (2008) state that the lack of support from the top management acts as a barrier to the formation of agile teams. Advocate role played by the agile coach who is identified in our research solves the above issues. Our research supports enhanced communication and to voicing views freely when allowed to work on an open space environment and trust that the top management have with the employees make the agile teams succeed well in the organization.

Downes (2007) describes about the open educational resource. It deals with learning through the available public resources. Our research suggests that the skill level of team members should match with the industry demands and they should realize that they gain more knowledge from real time training than the classroom training. They should know the difference between tacit knowledge and explicit knowledge.

Agile development approach is not document free. de Souza et al (2006) claims that software architecture documents are considered least importance among software maintainers. Berglund and Priestley (2001) favours the software documentation practices to "User driven" and "Just in time" writing. Our research also focuses on the feature requisite documentation rather than an exhaustive documentation as in traditional software projects.

Moe et al (2008) claims that lack of support from the top management is an obstacle in the way of an open independent way of working among the team members. Our research illustrates the shift in emphasis between liberty and responsibility makes the team member recognize their stand and involvement in their projects. Dyba and Dingsoyr (2008) have studied the balance between the team member's independence. Sharp and Robinson (2004); Whitworth and Biddle (2007) claim the responsibility and ownership feeling of a team member goes well with self-monitoring practices. Our research suggests that usage of report meetings and information radiators such as story boards and burn down chart make a team member pick a task of his own from the story board and self monitor his progress. His participation in monitoring team progress through burn down charts enables realization of his own responsibility through completion of the self-assigned tasks.

Derby and Larsen (2006) have identified the usage of retrospective as a way to self evaluate the team's performance. Our research states that the usage of retrospectives improves learning skills through collective practical experience. Rotation of the team members' roles make them conversant with the concepts and procedures in activities closely connected with their own functional area of strength making them generalists.

Tolfo and Wazlawick (2008) studied the influence of organizational culture on the adoption of XP. Their study concludes that while XP assumes an environment that is conducive for XP teams.

In depth communication with team members can be achieved with the help of working in open space environment. Fresh developers feel the pressure when working on new projects (Dagenais et al 2010). The presence of an agile coach in motivating a fresher will surely ease the pressure on him and will help the projects. Our research suggests the rotation with the team enhances the knowledge level of an individual thereby becoming a generalist.