ISSN No. 0976-5697

Volume 3, No. 5, Sept-Oct 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Transition from Classic/Traditional to Agile: Critical Steps for Smooth Transition

Lipika Bose
Software Engineer, HCL Technologies
Noida, India.
lipika.bose@gmail.com

Abstract : The Internet economy has altered the rules of software engineering. Traditional development methodologies are too cumbersome to meet the rapidly changing requirements and short product cycles demanded by business. This paper introduces critical steps required for smooth transition from classic software development model to agile methodology as per my experience in service organization such as HCL Technologies that needed to learn agile methodology to support the transition. It also lists the popular methodologies, metrics that matter and agile estimation and planning techniques. Finally, it summarizes the whole paper.

KEYWORDS: Classical Model, Agile Model, Transformation, Agile Methodologies, Metrics, Estimation And Planning.

I. INTRODUCTION

The day comes when suddenly somebody in the organization starts talking about Agile, and decides to implement Agile. This senior person who made this decision, would have heard somebody saying:

Agile improves productivity!! and saves money Or the customer would have said, if you don't practice Agile I will not give the project to you(in an outsourced scenario).

The planning of traditional software [1] development methods is conventional; it often defines a specific project scope firstly, then operates the top-down function decomposition based on the scope and estimates development time and development costs. In the agile development [2], one big software project is decomposed into multiple executable sub-projects. Firstly they finish the most important functions which are selected by users, use iterative incremental development methods, make each iteration results obtain a running system .Development team focuses on quickly dealing with the changing needs. When demand changes, development team quickly adjusts scheme. Because of quickly responding to user demand change, it is welcomed by small and medium-sized enterprises after it is put forward.

II. GLIMPSE OF CLASSICAL AND AGILE:

a. ***Classical Model:*** The classical/waterfall model[1] is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement Specification, Design, Construction, Testing, Production/Implementation and Maintenance. Thus the waterfall model maintains that one should move to a phase only when its preceding phase is completed and perfected. The idea behind the waterfall model may be "measure twice; cut once." Time spent early in the software production cycle can lead to greater economy at later stages. McConnell shows that a bug found in the early stages (such as requirements specification or design) is cheaper in money, effort, and time, to fix than the same bug found later on in the process. Those who opposed to the waterfall model argue that this idea tends to fall apart when the problem constantly changes due to requirement modifications and new realizations about the problem itself.

b. ***Agile Model:*** Agile software development [2] is a group of software development methods based on iterative and incremental development. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames (timeboxes) that typically last from one to four weeks. Each iteration involves a team working 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 minimizes overall risk and allows the project to adapt to changes quickly. Stakeholders produce documentation as required. An iteration might 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 might be required to release a product or new features.

III. TRANSFORMING FROM THE CLASSICAL MODEL

The transition from waterfall to agile software development requires careful planning, collaboration and change management. When implemented correctly, the agile model can result in your project team accelerating production efforts and working more cohesively [3].

Below are the vital points to consider for smooth transition to agile [3][4]:

a. ***Expect Some Bumps In The Road - Change Is Never Easy [5]:*** Remember people will always resist any new change. People are happy with their comfort zone. So let them take some time to realize that they will not have any choice but to uncover the new ways of the development. A good tactic would be to set-up a meeting room or some office area dedicated to documentation and planning for the agile transition. Post some white papers on the wall, perhaps some diagrams of the new process - whatever will help get the team thinking, breathing and offering solutions for the transition.

b. ***Training Required:*** Get all the team members to undergo training on Agile (XP, Scrum, etc). Entire team has to attend the agile training [5][6]. I would suggest starting the training from the top. As per my understanding, this initiative will be a successful one if the top management understands the agile. Middle managers should go for agile certification. Developers and testers can attend at least 8 hours of agile training to start with. Otherwise, agile is not that easy to learn in 8 or 16 hours. Every person will need to read a lot before practicing at the floor. Give them their own time to learn the agile. They will take time to wear the new skin of the agile development.

c. ***Agile Coach Needed:*** Even when the team is starting with just a few practices, handholding/mentoring from an agile coach [6] is needed. A seemingly simple practice like standup meeting also needs guidance in the initial stages. Make a group of expert people who will provide the rules and regulations, policies, guidelines for the project. You can call this group anything. Traditional companies call them Agile PMO or Agile COE etc. Remember these people need to be expert. If any company does not have these people then hire them. These people will be pillars in this initiative.

d. ***Story Board saves time:*** It is advisable to have a story board which would be helpful in tracking stories and disseminating the information to the entire team. The board can have the following columns: Not Started, Analysis, Design, Coding + UnitTesting, System Integration Testing, User Acceptance Testing, and Released.

e. ***Co-ordination:*** Teams need to coordinate during all planning meetings [7]. Especially important are the release planning meetings, where milestones/ deliverables are determined by agile project managers and teams. Continued follow-up and adjustments are covered in iteration planning meetings and daily standups. Person in rotation from the team can be selected as the coordinator, who would focus the team to talk about what was done yesterday, plans for today and problems encountered.

f. ***Consideration of few practices at a time:*** Don't force all the Agile practices [6][7] at once. Take one or two practices at a time and give sufficient time for the team to learn and practice. - For ex: One can start with shorter iterations and scrum/daily stand up meetings.

g. ***Initial longer iteration period:*** Start with 4-6 weeks iteration rather than 1 week iteration. Many new comers to Agile feel suffocated with 1 week iterations. Earlier the waterfall teams would have delivered softwares once in 6 months, and suddenly asking them to deliver in a short period makes them resist to Agile. Start with the 4 weeks sprint cycle. As per my experience, developers and testers prefer the longer duration. They would prefer 4 weeks sprint, which is also quite good.

h. ***Initial dismantling of current team is not favourable :*** Scrum and XP [8][9] advocates specific team structure like having Product Owner, Team, cross functional teams, no hierarchy, a team coach, scrum master, etc. This is a very sensitive issue. Suddenly informing the team that all of you are same, might heart the ego of senior people. So, better not to worry about dismantling the current team structure [5]. Let the team learn slowly the importance of the values and decide what is best for them.

i. ***Different financial reconciliation practices required:*** Agile methodology also requires different financial reconciliation practices. Because teams will go through more iteration, and potentially even change some of the original project specifications, the Project Manager will have to implement more check-points to assess budget. To exercise budgetary control, the Project Manager must assess and dole out project hours to team members in smaller chunks – hours should be assigned to each team member weekly.

j. ***Establish a Rhythm of Inspection and Adaptation***: In the review and retrospective held at the end of each iteration, analyze the benefits and challenges you've just experienced and make recommendations on how to improve the experience in the next iteration.

k. ***Test updated as design/coding progress:*** Efforts should be made to strengthen test driven development. Tests should be written upfront and would be updated as design/coding progressed [3][4]:. The Product Owners and Developers also play a big part in achieving agile testing. The Product Owners needs to participate in defining the acceptance tests prior to the sprint planning meeting.

l. ***Automated testing tool preferred:*** QA needs to get working builds from development as needed. Testing needs to start day 1 of the sprint. Automated testing tools [7] is less time consuming. Everyone owns the quality of your software, not just the QA team members. Every member on an agile team plays a significant part of ensuring the quality of software. Testing begins long before the testers start writing automated scripts.

IV. IMPORTANT AGILE METHODOLOGIES

There are many agile development methods [3][8], Extreme programming is one of the most successful practices in agile development.

A. ***Extreme Programming:*** Extreme programming [8] is invented by K.Beck. Extreme Programming is successful because it stresses customer satisfaction. It advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. It begins from collecting user stories, chooses valuable user story into iteration planning,uses test-driven development way, and then puts them into the integrated repository after tests.

a. ***It is implemented in the following ways:***

a) ***Paired Programming:*** Pair programming [8] is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer reviews each line of code as it is typed in. The two programmers switch roles frequently. While reviewing, the observer also considers the strategic direction of the work, coming up with ideas for improvements and likely future problems to address.

b) ***Test Driven Devlopment***: Test-driven development (TDD) [8] is an evolutionary approach to development which combines test-first development where you write a test before you write just enough production code to fulfill that test and refactoring. What is the primary goal of TDD? One view is it's one way to think through your requirements or design before your write your functional code. Another view is that TDD is a programming technique. The goal of TDD is to write clean code that works.

c) ***Continuous Integration:*** Continuous integration [8] aims to improve the quality of software, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control after completing all development. Continuous integration refers to integrate the finished function modules into the code library, then execute compile process and run all testing. Continuous integration can detect defects early, hence it can shorten time to market, increase the transparency of project.

Extreme Programming has many practices like reconstruction, daily meetings etc.

B. ***Scrum:*** The main roles of Scrum [9] are: the "Scrum Master", who ensures the process is followed, removes impediments, and protects the Development Team from disruption, the "Product Owner", who represents the stakeholders and the business , the "Development Team", a cross-functional, self-organizing team who do the actual analysis, design, implementation, testing, etc. Scrum projects make progress in a series of sprints, which are timeboxed iterations no more than a month long. At the start of a sprint, team members commit to delivering some number of features that were listed on the project's scrum product backlog. At the end of the sprint, these features are done --they are coded, tested, and integrated into the evolving product or system. At the end of the sprint review is conducted during which the team demonstrates the new functionality to the product owner and other interested stakeholders who provide feedback that could influence the next sprint.

C. ***Feature Driven Development:*** There are five main activities in FDD [3] that are performed iteratively. The first is Develop An Overall Model. At the start of a project your goal is to identify and understand the fundamentals of the domain that your system is addressing, and throughout the project you will flesh this model out to reflect what you're building. The second step is Build A Features List grouping them into related sets and subject areas. Next you Plan By Feature, the end result being a development. The majority of the effort on an FDD project, roughly 75%, is comprised of the fourth and fifth steps: Design By Feature and Build By Feature. These two activities are exactly include tasks such as detailed modeling, programming, testing, and packaging of the system.

V. METRICS THAT MATTER

Metrics [10] is used to detrmine the status of a project and are ways by which a project manager enables delivery teams to see where resources are needed or spent, or which areas of a project need more focus. There are various metrics that provide the measure of the state of an agile project. Some of them are listed below.

a. ***Progress:*** Progress is a function of converting requirements into a working version of software. The initial startup step of progress is plan. Plan involves the estimation of effort required for specific requirements and the actual value. Comparing the plans and the efforts allows the manager to adjust their metrics for next iterations and requirements. It allows sponsors to have a good idea of the amount of resources to assign to the project team, as well as agree on project milestones.

b. ***Quality:*** Software functional quality [10] reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly. Structural quality is evaluated through the analysis of the software inner structure, its source code, in effect how its architecture adheres to sound principles of software architecture. In contrast, functional quality is typically enforced and measured through software testing. Identifying the defects at the earlier stages of the project is one of the key strengths of agile methods, therefore quality checks prevent further rework throughout the progress of the project. It is important, however, that new test cases should be included in every iteration or when changes are introduced to the system. Counting the total number of test cases vs. open defects reflects the progress that the team has made in terms of development work.

c. ***Team Morale:*** The higher the morale, the more effective team members work; the more attrition is prevented, overall performance is reflected in the product's quality. Overall team morale is directly related to the level of individual morale. The amount of personal investment that people put into the project is related to how they manage stress levels as the next iteration approaches, and how issues are communicated during development or rework.

d. ***Effort Overrun:*** Now when sprint is over, you need to compare the planned effort (from available capacity) Vs the actual efforts. What is the acceptable percentage in the variance? It depends on your project or company.

e. ***Capacity Utilization:*** How much time are you spending in the productive delivery/engineering activities?

f. ***Leave Details:*** This is very important in Agile projects as you have only limited time to deliver the sprint commitment.

VI. AGILE ESTIMATION AND PLANNING

Estimating is estimating the [resources, time, size] required to develop a [user story, feature, or requirement]. Planning is putting the estimates together to formulate a project plan and schedule.

Estimations cab be made at three levels[11]:

A. Iteration Plan Estimation.
B. Release Plan Estimation.
C. Project Estimation.

Some of the estimation units are:

a) Ideal Time: Time taken to complete the task if there are no interruptions.
b) Story Points: Relative measurements among user stories like user requirements, feature lists, use case scenario.
c) Velocity: Number of estimation units that get completed by a team in single iteration.

a. Iteration Plan Estimates: In this the entire team gets together at the beginning of an iteration. They prioritize each of the items. Estimation of each item is done. Use the team velocity to stack the tasks into iteration. Velocity is estimated using historical data ie. by previous iterations or by running the average of past iterations. Velocity [3] is also calculated using forecasting such as determining the ideal hours per iteration. Burn down charts are prepared showing amount of work remaining at the start of each iteration.

b. Release Plan Estimation: The goal of initial release planning [11] is to estimate roughly which features will be delivered by the release deadline (presuming the deadline is fixed), or to choose a rough delivery date for a given set of features (if scope is fixed). It involves identifying and commting to the following[11]:

A goal for the release.

A prioritized set of stories that will be developed in the release.

A coarse estimate for each story.

Project managers, senior developers and business analysts involved in release planning or estimation. It is similar to iteration planning but here we extend the technique to multiple iterations. In agile projects we plan continuously, and we correct our course as we go. One of the primary mechanisms for course correction is allowing the release plan to evolve in response to all kinds of feedback. It will take at least a couple of iterations for team velocity to settle down. Iterations will sometimes deliver less functionality than was planned for, and sometimes more.

c. Project Estimation: The key factors of project estimation are cost, effort, time. Planning poker is one the common estimation technique in agile. In this each estimator is given a card. Each card contains a valid estimate. Story is read and discussed briefly. Each estimator selects a card that reflect their estimates. Discussions take place and re-estimations are done to get convergence.

VII. CONCLUSION

Thus by following some above mentioned steps transformation from a waterfall model to agile can be simpler. Different methodologies depending upon the project requirements are followed. Proper estimation and planning plays a vital role in the success of releases. Various metrics are kept in consideration to improve the overall quality of the project.

One common criticism of agile software development methods is that it is developer-centric rather than user-centric. Agile software development focuses on processes for getting requirements and developing code and does not focus on product design. Agile methods seem best for developmental and non-sequential projects. Many organizations believe that agile methodologies are too extreme, and adopt a hybrid approach that mixes elements of agile and plan-driven approaches.

VIII. REFERENCES

[1]. Humphrey W., “A Discipline for Software Engineering", Addison-Wesley, 1995.

[2]. See A. Cockburn, “A Methodology Per Project,” 1999, http://alistair.cockburn.us/crystal/articles/mpp/methodology perproject.html and A. Cockburn, Agile Software Development, Addison Wesley, Boston, 2002.

[3]. Robert C.M, Martin, Micah Martin. Deng Hui, Sun Ming translation, “Principle, Mode and Practice of Agile Software Development”. Posts & Telecom Press. 2008. pp: 123-156.

[4]. Schatz, B. & Abdelshafi, I. Primavera gets agile: A successful transition to agile development. IEEE Software. 22(3). 2005.

[5]. Lawrence, R. Avanade Inc., Seattle, WA Yslas, B. Three-way cultural change: introducing agile within two non-agile companies and a non-agile methodology. Agile Conference, 2006. Conference Publications. 5pp.-262.2006.

[6]. Boehm, B. & Turner, R. Management challenges to implement agile processes in traditional development organizations. IEEE Software. 22(5), 30-40. 2005.

[7]. Livermore, J.A. Factors that impact implementing an agile software development methodology. Southeast Con, 2007. Proceedings. IEEE. Conference Publications .82-86.2007.

[8]. DaYong Sang, Wang Ying, LiHua Wu.”Agile Software Development Methods and Practices”Xi-an: XIDIAN University press.2010.pp:160-180.

[9]. Mann, C. & Maurer, F. A case study on the impact of scrum on overtime and customer satisfaction. Proceedings of the Agile development Conference (ADC’05). Denver, CO. 70-79. 2005.

[10]. Da Yong Sang. “Requirement Analysis of Agile Development Process “.Programmer, 2009, 2, pp: 70-75.

[11]. M. Cohn, Agile Estimating and Planning, Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, 2006, pp. 215-245.