ISSN: 0976-3376

Available Online at http://www.journalajst.com

ASIAN JOURNAL OF SCIENCE AND TECHNOLOGY

Asian Journal of Science and Technology
Vol. 07, Issue, 11, pp.3908-3911, November, 2016

RESEARCH ARTICLE

AN ANALYSIS OF CHALLENGES IN AGILE DEVELOPMENT

***Arvind Pillai and Mythili Thirugnanam**

School of Computer Science and Engineering, VIT University, Vellore, India

ARTICLE INFO

Article History:
Received 27th August, 2016
Received in revised form
17th September, 2016
Accepted 01st October, 2016
Published online 30th November, 2016

Key words:

Agile development, Risk identification,
Work breakdown structure,
Gap analysis, Effort estimation,
Challenge.

ABSTRACT

Software project management techniques and business methodologies have been developed from practices in the Information Technology industry to ensure a high rate of success with the given deadlines. These techniques have been adopted by almost all the top firms to ensure a lucrative and efficient solution to a problem. Moreover, the size of the project management team and model embraced to ensure success depends on the nature of the project. Agile development model using agile teams are one way of solving these complex problems in a flexible way, this development model is gaining popularity at a tremendous rate, especially among small companies comprising of limited members in a team. Therefore, this paper reviews the main challenges in an agile development environment and the methods employed to overcome these risks and challenges.

INTRODUCTION

Business applications are systems that are today typically accessed via Internet technologies. As per (Ghanbury, 2006) the technology has increased the connectivity all around the globe and organisations are globalising faster than ever before. This advancement has raised the expectations of people in their work conditions and living standards. From the perspective of (Unhelkar B, 2010), agile methods are developer centric, starting with the developer writing a part of the code and comparing it with the requirements of the user and demonstrating the working of the code, the user gives his feedback through which the code can be enhanced or modified. Agile approach is more efficient (Karlesky and Vanter October 2008) than the traditional approach of using the waterfall model. The main reason for this is the inability of the waterfall model to accommodate changes in the requirements when the project is nearing completion. From the waterfall model depicted in (Govardhan*et al*, September 2010), it can be observed that the user must specify the requirements to the user at the beginning of the first phase, and these requirements must be complete and accurate, the inability of the client to lead report all the requirements with maximum accuracy leads to the failure of this model. Therefore, to compensate for the limitations of the traditional approach for project management, major technology firms such as Yahoo and Google have adopted agile development model for some of their teams.

****Corresponding author: Arvind Pillai,***
School of Computer Science and Engineering, VIT University, Vellore, India

Highlights on some agile development methods

Crystal clear method (Cockburn, 2004) is used mainly to facilitate small team which are not life-critical and co-located teams having different size and different stages of criticality: Clear, Yellow, Orange, Red and Blue. It is the most agile method which has seven characteristics: frequent delivery, reflective improvement, personal safety, easy to access to expert user, focus, osmotic communication, and requirements for technical environment. Dynamic software development method (DSDM) (Stapleton, 2003) involves the division of projects into three phase namely, pre-project, project life-cycle, and post project. DSDM is composed of nine principles: empowering the project team, frequent delivery, high-level scope being fixed before project starts, testing throughout the lifecycle, user involvement, addressing current business models, iterative and incremental approach, allow for reversing changes and efficient communication. Feature-driven development (S.R. Palmer and J.M. Felsing, 2002) uses a mixture of model-driven and agile development with importance laid on the initial object model, iterative design for each feature and division of work in features. Moreover, design and development are the two phases of an iteration. This model can be used to develop critical systems. Lean software development (M. Poppendieck and T. Poppendieck, 2003) is developed from the principles of lean production, to be more specific, from the Toyota production system to software development. This is model is comprised of seven principles: amplify learning, decide as late as possible, empower the team, eliminate waste, build integrity, deliver as

fast as possible and see the whole picture. Scrum (K. Schwaber and M. Beedle, 2001) is a model that lays emphasis on project management in scenarios where planning ahead is difficult. This model consists of mechanisms for "empirical process control", the main idea of this concept revolves around feedback loops. Software is developed by a self-organizing team in increments (called ''sprints"), starting with planning and ending with a review. Features to be implemented in the system are registered in a backlog. Then, the product owner decides which backlog items should be developed in the following sprint. Team members coordinate their work in a daily stand-up meeting. One team member, the scrum master, oversees solving problems that stop the team from working effectively. Extreme Programming (Beck, 2000) (Beck, 2004) sheds light on the best practices which can enchance development. It is composed of twelve practices: collective ownership, small releases, simple design, on-site customers, 40-h week, coding standards, the planning game, pair programming, metaphor, refactoring, testing and continuous integration. XP2 is a revamped version of extreme programming which consists of the following practices: whole team, continuous integration, 10-minute build, slack, test-first programming, stories, pair programming, weekly cycle, quarterly cycle, sit together, informative workspace, energized work and incremental design. This model gives a supplementary information about eleven corollary practices.

Challenges in agile development

There are several fundamental differences between traditional and agile models as shown in Table 1. This gives rise to several challenges in agile development.

Table 1. Difference between Traditional and Agile project management models

Domain of difference	Traditional	Agile
Core idea	This model is dependent on processes and the complementary tools.	This model is dependent on the cooperation between people in a team.
Documentation method	Work documents must be filed which measures each activity.	Software is used to measure the progress of the project and more interested in the design code.
Interaction	Client-Developer interaction is minimal.	Client-Developer interaction is an integral part and it takes place frequently.
Flexibility	It is rigid, i.e. all requirements must be specified before the end of the first phase.	It is more flexible, i.e. changes to the project can be made in almost any of its phases.
Extradition	Does not permit the extradition continuously working for the client.	Permits the extradition continuously working for the client

From Table 1, It is understood that agile development lays emphasis on the psychology and behavioural aspects of human beings. However, because of the unpredictable nature of these aspects we need to make certain modifications to the traditional model to extract the full potential of agile development. Therefore, issues and solutions with respect to risk, work breakdown structure, gap analysis and effort estimation are discussed in the following sections.

Risk identification in agile environment

In an agile environment risk is depended upon the two intrinsic factors namely, Validation and Verification. Risks are identified to manage them properly using a risk management plan, the following risks are crucial to a project with uses agile development:

- Validation makes sure that the system built is in accordance with the requirements specified by the user.
- Verification analyses the outputs given by the system for multiple iterations and verifies the output through a process of functional testing and code reviews.
- Human threats such as lack of human resources to complete all the tasks within the deadline, unexpected holidays due to medical reasons by the members of the team or death of a family member, this threat also accounts for lack of harmony within a team.
- Operational threats such as the lack of sufficient resources for each team.
- Political threats such as change in national policies and compliance agreements because of a newly elected government.
- Financial factors which include both internal and external sources of monetary funds.
- The cost and time constraints that determine the success of a project.
- Technical factors: a piece of technology might be obsolete when the project enters the highly competitive market.

Work breakdown structure in agile development

The traditional work breakdown structures may not be completely compatible with agile projects because agile involves constant interaction with the client through the entire course of the project. In an agile environment, the ideas suggested by the client must be implemented as and when it is communicated with the team. Therefore, we follow certain rules or techniques while constructing our work breakdown structure to ensure successful and satisfactory completion of the project. The features that needs to be implemented must be placed in a forced rank order, i.e. each feature must be given a priority number which signifies the lowest number has highest priority, to be specific, a task or feature assigned the priority number '1' must be executed first before proceeding to the other task. The use of priority words such as 'Maximum', 'Minimum', 'Most', 'High', 'Moderate', 'Low' must be avoided (Bozzuto and Brian, 2011).

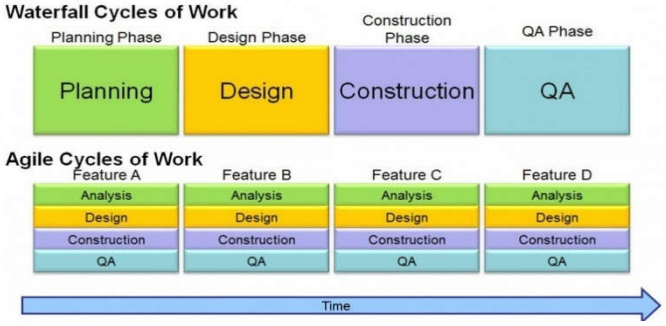

Fig. 1. Difference between traditional and agile cycles of work (Bozzuto and Brian, 2011)

In Fig. 1, it can be observed that unlike the traditional approach the agile development model uses analysis, design, construction and QA phases for each feature, the priority of a feature can be changed during the project provided the feature has not yet started. In an agile environment, we divide the work in to smaller pieces which can be completed in under two weeks, this is done to improve testability and flexibility of individual features. Work breakdown structure is usually constructed by the project manager or other technical heads who usually think from a one-dimensional perspective, but the project is carried out for the convenience of the user. Hence, we must this from a user's perspective, this is usually achieved by a concept from extreme programming. User Stories is a part of extreme programming which consists of two components: a narrative and acceptance criteria. The narrative contains an actor, an action and a justification and the acceptance criteria contains the criteria the business will test to approve a given story.

Table 2. Narrative and Acceptance Criteria for a multiplayer game with priority

Priority	Narrative	Acceptance Criteria
1	1. As a gamer, I want to search for heroes using by their names, so that I can select the hero I want to play	1. I can search based on an exact hero name. 2. I can search based on letters within the name of the hero.
2	1. As a gamer, I can select a list of five heroes for my team, to build a list of heroes for my team	1. I can add heroes from my search list. 2. Delete heroes from my search list. 3. Submit my selected team. 4. Ban heroes that my opponent cannot select.

The information in table 2 gives us an example of narrative and acceptance criteria for a multiplayer game, this will give us further insight into the solution.

Gap analysis in agile development

Gap analysis is the comparison of actual performance of an information system with the desired performance which will improve efficiency. It consists of a current state which gives us information about the current performance exhibited by the system, future state which informs us about the performance which is expected, gap description which will outline the factors which contribute to the gap and steps and proposals to bridge this gap.

Table 3. Difference between agile and traditional gap analysis

Agile gap analysis	Traditional gap analysis
Gain an understanding about the actual process used by people.	Read through gap analysis documents to check whether the documented processes are followed.
Requires rigorous analysis and multiple Q&A sessions with the members of the team.	Does not provide insight on the real processes followed by the people.
It helps us uncover the most critical gap which can be improved to increase performance.	Behaviour change is very difficult to improve because it does not involve constant interaction with the members of the team.

There are several ways to run a gap analysis, but in an agile environment we will focus on the behavioural route instead of the rigid documentation route. Table 3 illustrates the differences between an agile gap analysis and a traditional gap analysis (McMahon and Paul, 2010).

Table 4. Criteria for size estimation

Size	Criteria to assign this category
Small	Single task at a location must be executed only once.
Medium	Few tasks at multiple places must be executed few times.
Large	Many tasks at multiple places must be executed repeatedly.

Table 5. Criteria for complexity estimation

Complexity	Criteria to assign this category
Low	The knowledge and skill needed to accomplish this task is already known (or) this task has already been done before.
Medium	This is not familiar but with reasonable amount of research the task can be accomplished.
High	This task has neither been done before nor been accomplished by anyone, people lack the knowledge need to accomplish this task

In an agile gap analysis, interviews are conducted mostly in an informal manner to keep the interaction as comfortable as possible, specific words in the CMMI (Capability Maturity Model Integration) model are not used to keep the discussion focused on the team member, some questions which are asked in the interview by the project manager are as follows:

- "How do you perform your work?"
- "Do you use a specific process or randomly start your work?"
- "Do you document your progress?"
- "Do you consider yourself to be efficient in your work?"

The project managers usually lay emphasis on listening during the interaction, this interaction is recorded and later analysed by the managers, the analysed evidence is then compared with the processes listed in the CMMI model, to be more specific, the processes which are omitted by the team member or the processes which go against the CMMI model. Consequently, a detailed report is generated and given to the senior management of the organisation or the client.

Effort estimation in agile development

Effort estimation is an important task in software development which determines constraints such as time, human resources and money. Traditional effort estimation techniques such as COCOMO which are used to determine entire lifecycle processes cannot be used in an agile environment because methods used in agile programming like Scrum and Extreme programming are iterative processes which focus on many processes with small complexity. The reasons why we cannot use a traditional approach in agile environment are listed below:

- Traditional methods will not work for large amounts of data which are pooled in from similar projects. Agile development focuses on diversity of data.

- Incremental approach does not focus on listing all the requirements in the initial phase followed by figuring out the entire lifecycle time, therefore we cannot use traditional methods.

Let us consider Extreme Programming, a planning game is used to determine the nature of iteration. The effort is estimated is estimated by the developers in units of effort rather than in units of time. Project velocity is a method proposed in (Jeffries *et al*, 2001) to determine the number of units of effort that can be done in a week. A method used to calculate the effort in an agile environment is proposed by (Fernando Machado and Luis Joyanes, 2005) which uses a 4-step procedure:

- The size and complexity of each task is estimated using a set of guidelines.
- Classify the tasks based on size and complexity described in table 4 and table 5.
- Depending on the category of a task, the number of hours required for this task can be calculated using the rate of that category, the rate varies with respect to each iteration.
- Finally, all the hours computed are added to form a single entity which is then divided by the number of developers assigned to the iteration to determine the duration of the entire iteration.

Conclusion

To conclude, from this paper it is clearly understood that agile development is different from traditional methods in several different ways, moreover, although agile developments has certain limitations related to the human resources and psychology of humans it has proved itself worthy in small teams such as start-ups. Furthermore, several novel methods are being proposed to facilitate the infusion of agile development into mainstream companies, these methods are related to improving the efficiency of effort estimation, better understanding of risks involved in an agile environment, giving better techniques to project manager to handle their teams and improved gap-analysis techniques.

REFERENCES

Beck, K. 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley.

Beck, K. 2004. Extreme Programming Explained: Embrace Change, second ed. Addison-Wesley.

Bozzuto, B. 2011, September 14). The Product Backlog, an Agile WBS. Retrieved from http://www.solutionsiq.com/product-backlog-agile-wbs/

Cockburn, A. 2004. Crystal Clear: A Human-Powered Methodology for Small Teams. Addison-Wesley.

Fernando Machado, L. J. 2005. Effort Estimation in Agile Software Development: A method and a Case Study. Software Engineering Research and Practice, 470-475.

Ghanbury, A. 2006. Collaborative Business Process Engineering across Multiple Organisations. ACIS.

Govardhan, N. M. (September 2010). A comparison between five models of software engineering. *International Journal of Computer Science* Issues, Vol. 7, Issue 5.

Jeffries, R., A. A. 2001. Extreme Programming Installed. Addison-Wesley.

Karlesky, M., M. (October 2008). Agile project Management (or, Burning Your Gantt Charts). Embedded Systems Conference Boston, 247-267.

McMahon, P. E. (2010, August 9). Bringing Process Maturity to Agile Organizations. Retrieved from informIT: http://www.informit.com/articles/article.aspx?p=1620555&seqNum=4

Poppendieck, M., T. P. 2003. Lean Software Development- An Agile Toolkit for Software Development Managers. Boston: Addison-Wesley.

S.R. Palmer, J. F. 2002. A Practical Guide to Feature-driven Development. NJ: Prentice Hall.

Schwaber, K., M. B. 2001. Agile Software Development with Scrum. Upper Saddle River: Prentice Hall.

Stapleton, J. 2003. DSDN: Business Focussed Development, second ed. Pearson Education.

Unhelkar, B. 2010. Agile in Practice - A Composite Approach. Cutter Consortium, Vol 11, No 1.

****