AGILE: The Innovation with Intelligence

Jayanthan, R, Gowtham, R, Praveen T, Sivakumar P, Sankar G

1Dept. of CSE, Veltech High-Tech Dr. Rangarajan & Dr. Sakunthala Engineering College, Avadi, Chennai, Tamilnadu, India
2Dept. of CSE, Amrita Vishwa Vidyapeetham University, Coimbatore, Tamilnadu, India

Abstract
“AGILE” is the most powerful project model which tends to grouping of cross functional team workforce and also the continuous integration of the process which over and done with the successful project delivery. We was following the traditional waterfall model and V-shape in our organization and once we started a project with AGILE where this is a project with 4 sprint and total of 30 days for project deadline. In this model the development and Quality Control (QA) team will work together and in scrum call the estimation was did based on various resources around the geographic locations, so the planning will made in accurate level for the development because the estimation is done by multiple brains and finalized after discussions and also when the user develops with programming the QA team will prepare the necessary test cases so which can be shared with the developers so which may discard the issues while coding.

In the QA team the Tester will get a chance to test the entire nook and corner of the functionality areas since each user stories are small so everyone get a fair knowledge on business level, so at the end the code delivered will be error free and perfect. The scrum call on daily basis helps everyone in the team to get update and fair knowledge of status, so if any one part face any blocks or huddles which can be owned by the expert of the team and completed on time. This method helped us a lot for the proper delivery of the project on time with 100% Accuracy and functionality as expected.

Keywords
Agile, Project Management, Scrum, Advantage of Agile, Project Delivery

I. Introduction
In this paper we are going to get knowledge on the agile process and also which also help everyone to know the following and get a better idea on Agile methodology.

- What is Agile?
- Is Agile helping our organization?
- Can Agile deliver a perfect product?

We can also get a clear view on the Agile and how it differ from the traditional Waterfall Model and Impact of following Agile in the organization for the project implementation and also the various surveys and case studies scenario which proves how this new methodology helped for the right delivery of product on right time.

II. Principles of AGILE
The major principle of agile manifesto contains the twelve golden parameters to follow during the phase of project development.

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

III. AGILE Methods
All agile methods are described in (Abrahamsson, 2002) as displaying the following attributes,

- Incremental development: small software releases with rapid development cycles.
- Cooperative development: close customer and developer interaction.
- Method simplicity: easy to learn, modify and document.
- Adaptive development: simple and effective change management at any point within the overall software life-cycle.

IV. Extreme Programming (XP)
Extreme Programming (XP) is the most widely recognized agile method. XP has been pioneered by Kent Beck and is described in as “a light-weight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly-changing requirements”. XP originated as a prototypical C3 payroll system development project within the Daimler-Chrysler organization. XP is based on four values and an initial set of twelve practices. The four values are as follows:

- Communication: Most project problems occur because of poor communication – therefore XP strongly promotes communication in a positive fashion.
- Simplicity: Develop the simplest product that meets the customer’s needs.
- Feedback: Developers must obtain and value feedback from the customer, from the system, and from each other.
- Courage: Be prepared to make hard decisions that support the other principles and practices.
V. SCRUM

Building complex products for customers is an inherently difficult task. Scrum provides structure to allow teams to deal with that difficulty. However, the fundamental process is incredibly simple, and at its core is governed by 3 primary roles.

- Product Owners determine what needs to be built in the next 30 days or less.
- Development Teams build what is needed in 30 days (or less), and then demonstrate what they have built. Based on this demonstration, the Product Owner determines what to build next.
- Scrum Masters ensure this process happens as smoothly as possible, and continually help improve the process, the team and the product being created.

VI. DSDM

Dynamic Systems Development Method (DSDM) is one of the many methods that fall under the Agile-umbrella. While this type of methodology is more highly used in the European area of the world, it’s slowly being integrated into more Agile projects over here in the states.

During the recent Agile Business Conference, Keith Richards gave a talk on DSDM and what it means in the Agile community of practice. It is interesting to note that there was a lot of comments regarding Scrum as more product-centric and DSDM as more project-centred.

“Projects deliver on time. The approach creates ownership. It”s easier to implement and support solutions. But it does take some work to get right. While the method is holistic, you need to be able to apply it in an intelligent way.”

A. Benefits of DSDM

“VII. AGILE Application Lifecycle

The Fundamentals of Our Agile Application Lifecycle Approach are following

B. Deploy New Functionality Regularly

New functionality deployed every 2 weeks

VIII. Key Aspects of AGILE Platform

There are major five key aspects are followed in the agile methodology and they plays the vital role in the delivery and how good to achieve the task on time with the quality factor.

1. Integrate with Databases & 3rd Party Systems
   - Easy to integrate at any time of project development in the sprints
2. Develop, Assemble & Reuse Components
   - Component library built for reuse
3. Deploy New Functionality Regularly
   - New functionality deployed every 2 weeks

IV. Table 1:

<table>
<thead>
<tr>
<th>Flexible Time boxes</th>
<th>Ineffective Daily Scrum meetings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delaying bug fixes</td>
<td>Poorly structured Backlogs</td>
</tr>
<tr>
<td>Unclear definitions of done</td>
<td>Ambiguous or ineffective requirements</td>
</tr>
<tr>
<td>Measuring effort rather than achievement</td>
<td>Uncommitted Scrum Team members</td>
</tr>
<tr>
<td>Subconscious trending towards waterfall</td>
<td>Ad-hoc work requests</td>
</tr>
<tr>
<td>Unproductive retrospectives</td>
<td>And many others</td>
</tr>
</tbody>
</table>

“There is a need to differentiate between the general term „agile”, the phrase „agile methods” and the concept of „agile project management.” In the UK there are only really 3 well known „agile methods” (XP, SCRUM, DSDM). Only DSDM can be used „as is” for projects. Scrum and XP are product delivery techniques they have no concept of „a project” as defined by approaches like PRINCE2.” – Keith Richards Interesting points here. In summary from Keith’s talk by Elizabeth Harrin:

- DSDM is different from traditional project management – You agree the time and the cost and then vary the scope accordingly to make sure you hit those targets.
- DSDM approach to time and budgets makes Agile a very cost-effective - But you run the risk of not getting everything you want from the project first time round.
- DSDM uses a lot of facilitated workshops to improve communication flow – The people involved in the project and help move things forward quickly.
4. Manage Process, Team & System
- Agile Platform’s Service Centre sent real time alerts of any slow-down or bottleneck
- Agile Network used for project management & issues management
5. Enable Rapid Change
- Embedded Change based on feedback

IX. AGILE in Geographical Approach
Everyone has the doubt about how to implement the agile in a project or product which may be carried out by the team members which are segregated by the geographic locations around the world.

In the traditional model the requirement, analysis and design part will be done already and all the task and work breakdown structure will be ready to do assign task and kickoff the implementation of the project for the developers and also for the QA team members to test the project.

X. Spring Planning Meeting

The major strength of the agile is the communication, where all the members of the team are treat equally and given freedom to open their view on the project and the task while planning and estimating the user stories, daily communication will occur which will connect the members of the team in a single platform where they discuss about the process in the daily basis called the scrum call.

XI. Scrum Meeting

This is a 15 minutes call which will happen on the daily basis where the members of the team will discuss only the three things,
- What happened today?
- What will be done tomorrow?
- Any blockers or huddles?
This call help everyone to get clarity information of the process and the task that is happening in the project and which can also help the scrum master and the product owners to decide the exact date of delivery of the project or the product.

XII. Break Down Structure of Project
In an agile project, the high-level architecture is determined up front, and all the iterative designs fit into this architecture. The design at the end of an agile project is often substantially different than that at the end of a more process-intensive project. The architecture may also change as greater insight is gained during the solution life cycle. The Project Breakdown structure may look like below:
XIII. Traceability Matrix of Project

A traceability matrix is a document, usually in the form of a table that correlates any two baseline documents that require a many to many relationship to determine the completeness of the relationship. It is often used with high-level requirements (these often consist of marketing requirements) and detailed requirements of the software product to the matching parts of high-level design, detailed design, test plan, and test cases.

A requirements traceability matrix may be used to check to see if the current project requirements are being met, and to help in the creation of a Request for Proposal, various deliverable documents, and project plan tasks.

XIV. Survey for AGILE

There is growing survey evidence that agile works better than traditional. The DDJ 2007 Project Success Survey showed that when people define success in their own terms that Agile projects had a 72% success rate, compared with 63% for traditional and 43% for offshoring. These figs. are summarized in fig. 8. The DDJ 2008 Agile Adoption Survey showed that people’s experience with agile software development was very positive, see fig. 4, and that adopting agile strategies appears to be very low-risk in practice (few organizations seem to be running into serious problems, and most are clearly benefitting). For the 2010 IT Project Success Rates Survey we adopted some ideas from the Chaos Report and starting talking in terms of success, challenged, and failed. Results from the 2011 IT Project Success Rates Survey Results are depicted in fig. 4 and as you can see agile still outperforms traditional approaches.

There is growing survey evidence that agile approaches work as well or better than traditional approaches at scale. I’ve also run surveys which explored project success rates by paradigms at different levels of scale. At this point I’ve only explored two of the eight scaling factors of the Agile Scaling Model, in this case team size (see fig. 11) and geographic distribution (see fig. 12) but these two factors tend to be the ones people are most interested in. The July 2010 State of the IT Union survey explored the issue of team size and as you can see in fig. 11, AGILE approaches do as well or better than traditional approaches regardless of team size. The 2008 IT Project Success Rates Survey explored the issue of geographic distribution and as fig. 12 shows agile approaches do as well or better regardless of the level of team distribution.

Here is the result of the survey which compares various strategies of the deliverable like quality, productivity, customer satisfaction and cost of development.
Fig. 4: Effectiveness of Agile Software Development Compared with Traditional Approaches

This figure gives the clear view of comparisons of various project development methodologies, where the Agile small team size and best delivery of project.

Fig. 5: Comparing Paradigm Success Rates by Geographic Distribution

Shorter feedback cycles lead to greater success. Fig. 13 maps the feedback cycle of common development techniques (for the sake of brevity not all techniques are shown) to the cost curve. Agile techniques, such as Test Driven Design (TDD), pair programming, and Agile Model Driven Development (AMDD) all have very short feedback cycles, often on the order of minutes or hours. Traditional techniques, such as reviews, inspections, and big requirements up front (BRUF) have feedback cycles on the order of weeks or months, making them riskier and on average more expensive.

Fig. 6: Mapping Common Techniques to the Cost of Change Curve.

XV. Proof of AGILE Methodology Implementation

A. Case Study of a Company [Effective Soft]
Agile methodology can be applied to nearly any software project. It is especially beneficial for the projects the structure and features of which are changing during the process of development. One of the projects that were successfully implemented with Agile project management methods was Link Stock solution.

B. Problem
A large medical company was using the link exchange method while promoting their website. A special team of SEO specialists was searching the Internet for relative websites to exchange links with. The link base grew and needed close attention as each link had to be checked for validity on a regular basis. This check was manually done by the team members. At some point the marketing manager realized that the process of posting and checking the links could be optimized.

C. Solution
The company marketing manager suggested creating a special solution that would stock the links and check them regularly. The company decided to order the software development services at EffectiveSoft. As the requirements to the project were not completely formed from the start EffectiveSoft project manager chose to work on agile methods: after a certain period of time the customer was presented the project sprint-release. The customer in his turn presented their new requirements and ideas. The following Agile principles were observed while working at the project:
- Face-to-face communication was preferred to written documentation
- The team was located in the same office to provide better daily cooperation
- The customer took an active part in developing process
U.K. telecom giant BT Group adopted agile programming in 2005 and found that after some initial speed bumps, the development philosophy is a boon to its business. In 2005, BT Group began replacing an aging Unix-based phone-traffic monitoring system with a Web-centric architecture. The intent: To allow traffic managers to make quicker changes to switches and other physical devices to handle shifts in network loads -- on any point in the company’s vast telecommunications network -- without risking system overloads. The new system, rolled out in late 2005, has made the work of these phone-traffic controllers much easier for network load balancing; the system it replaced was difficult to upgrade. At that point, few people in the company even knew how the old system worked, says Kerry Buckley, a lead software developer in Ipswich, U.K., who worked on that project team.

But the most dynamic part of the development effort was this: The project was completed within the construct of BT’s nascent 90-day agile development cycle. Prior to the London-based Telco giant’s shift to an agile development methodology in 2005, it could take three to nine months for a third-party developer to gather specifications. Then the development itself could take up to 18 months or longer to complete, according to Al-Noor Ramji, CEO of BT Design and CIO at BT Group.

A traditional software-testing cycle, typically done after coding had been completed, would have prolonged the project by several additional months, says Ramji. The company’s shift to 90-day and often 30-day software iteration cycles is at least four times as fast, he says, meaning they could deliver the end product that much faster.

The central idea behind agile programming is to code quickly, test out what you’ve done, fix any problems and then move on. Although telecommunications companies haven’t historically been associated with progressive development approaches like agile, BT’s IT organization needed to speed up its system development cycles to help it deliver new mobile and other types of telecommunications services. BT’s shift to agile also meant that its 3,000-person global development organization would be working more closely with end-users. This was especially true during the requirements-gathering stage to better understand and meet user needs, says J.P. Rangaswami, managing director of service design at BT Design in London.

To help beef up its developers’ people skills and understanding of agile, BT put its programmers through a series of classroom and hands-on training sessions, says Rangaswami. The company has also recruited an unspecified number of IT professionals with agile experience from a variety of industries over the past two years who are helping to teach other developers who are relatively new to these disciplines, he adds.

E. Changing the Mind-Set

Even though BT’s shift from traditional waterfall development techniques to agile has led to significant productivity and business benefits, it didn’t happen overnight, nor was it easy for a company as massive and widespread as BT, says Ramji and Rangaswami. For starters, the company’s IT leaders had to break through some misperceptions among internal and external business customers that agile meant they could introduce frequent feature changes during the development cycle to suit their whims, says Ramji. Plus, the company’s shift to agile development was “more readily accepted” by senior executives and junior staffs, says Ramji. Middle managers were more skeptical about how it may impact them. The naysayers included IT infrastructure managers who had gotten used to having more formalized documentation for new software or enhancements being made to existing systems, says Rangaswami.

XVI. Acknowledgment

This Agile methodology helped us to get a clear view on project for all whomever participated in development of project. We want to convey my special thanks for our Team members who helped a lot and motivated to learn this process and being a beginners we also achieved a great success of delivering the product on time with 100% accuracy and quality. Survey Monkey, supported us for conducting the survey and collecting the report on survey and also the senior people who shared their experience and knowledge for preparing this.

References
[1] Ken Schwaber, Mike Beedle,“Agile Software Development with Scrum”.
[3] Ken Schwabe,“The Enterprise and Scrum”.
[10] Mary & Tom Poppendieck,“Implementing Lean Software Development”.
[12] Eric Evans,“Domain-Driven Design”.

Appendix

XP TWELVE KEY PRACTICES KEY PRACTICE

A. The Planning Game
A quick determination of the scope of the next software release, based on a combination of business priorities and technical estimates. It is accepted that this plan will probably change.

B. Small Releases
Produce a simple working system quickly, and then release new versions on a very short cycle.

Metaphor
Guide all development with a simple shared story of how the whole system works.

C. Simple Design
The system should be designed as simply as possible at any given moment of time.

E. Continuous Testing (or Test Driven Development)
Programmers continually write tests, which must be run flawlessly for development to proceed. Customers write function tests to demonstrate the features implemented.

F. Refactoring
Programmers restructure the system, without removing functionality, to improve non-functional aspects, simplicity and flexibility. Refactoring strongly focuses upon the removal of code duplication.

G. Pair-Programming
All production code is written by two programmers at one machine.

H. Collective Ownership
Any programmer can change any code anywhere in the system at any time.

I. Continuous Integration
Integrate and build the system every time a task is completed. It is a fundamental requirement to always have an up-to-date working prototype.

J. Forty Hour Week
Work no more than 40 hours per week as a rule.

K. On-site customers
A customer representative (i.e. a subject matter expert) works full time within the development team.

L. Coding Standards
Adherence to coding rules that emphasize communication via program code.

R. Gowtham received his B.E. degree in COMPUTER SCIENCE AND ENGINEERING from Jayam college of engineering and technology, Dharmapuri, Tamilnadu, in 2002, the M.E degree in COMPUTER SCIENCE AND ENGINEERING from KSR engineering college, Tiruchengode, Anna University, Chennai, Tamilnadu, in 2004, and he is pursuing Ph.D degree in computer science and engineering from Anna University, Coimbatore, Tamilnadu. He has 8 years experience in various top engineering colleges in India like (VIT-Vellore, Sri krishna engg coll Coimbatore...). He was a teaching assistant, assistant professor, senior assistant professor with Department of computer science and engineering, Amrita University Coimbatore, from 2010 to till now respectively. His research interests include data mining, semantic web with OWL, cloud computing, mobile computing, Web Security, Web Mining and software engineering.

P. Sivakumar received his B.E. degree in Computer Science and Engineering from Bharathidasan University, Trichy, Tamilnadu, India, in 1998, the M.E. degree in software engineering from Anna University, Chennai, Tamilnadu, in 2004 and he is pursuing Ph.D degree in computer science and engineering from Anna University, Coimbatore, Tamilnadu, India. He has 13.7 years of experience in various engineering colleges in India as an Assistant professor & Associate professor. At present he is working as Associate professor in Computer Science and Engineering Department, Veltech High Tech Dr. Rangarajan and Dr. Sakunthala Engineering College, Anna University, Chennai. His research interests include Artificial Intelligence, Distributed Computing, Grid Computing, Cloud computing and software engineering.

R. Jayanthan received his B.Sc. degree in COMPUTER SCIENCE from Vysya college, Salem, Tamilnadu, in 2006, the M.C.A degree in computer application from Sri Ramakrishna Institute of Technology, Coimbatore, Anna University, Chennai, Tamilnadu, in 2009, and the M.Tech degree in computer science and engineering from Veltech University, Chennai, Tamilnadu, in 2011. He was a teaching assistant, assistant professor, with Department of computer science and engineering, Veltech High Tech Dr. Rangarajan & Dr. Sakunthala Engineering College, under Anna University, in 2011 and 2012 respectively. His research interests include semantic web with OWL, cloud computing and software engineering.

P. Sivakumar received his B.E. degree in Computer Science and Engineering from Bharathidasan University, Trichy, Tamilnadu, India, in 1998, the M.E. degree in software engineering from Anna University, Chennai, Tamilnadu, India, in 2004 and he is pursuing Ph.D degree in computer science and engineering from Anna University, Coimbatore, Tamilnadu, India. He has 13.7 years of experience in various engineering colleges in India as an Assistant Professor & Associate professor. At present he is working as Associate professor in Computer Science and Engineering Department, Veltech High Tech Dr. Rangarajan and Dr. Sakunthala Engineering College, Anna University, Chennai. His research interests include Artificial Intelligence, Distributed Computing, Grid Computing, Cloud computing, and software engineering.
G.SANKAR received his B.E. degree in Computer Science and Engineering from Shri Angalamman college of Engineering and Technology, Bharathidasan University, Tamilnadu, India, in 2003, the M.E. degree in Computer Science and Engineering from Annamalai University, Tamilnadu, India, in 2005. He has 7.6 years of experience in various Engineering Colleges in India as an Assistant Professor. At present he is working as assistant professor, in Computer Science and Engineering Department, Veltech Hightech Dr.Rangarajan and Dr.Sakunthala Engineering College, Anna University, Chennai. His research interests include Artificial Intelligence, Distributed Computing, Grid Computing, Cloud computing, and software engineering.