SWOT Analysis of Agile Methodologies

Shubh
Assistant Professor
Department of Information Technology
Jagan Institute of Management & Sciences
shubh.aneja@gmail.com

Priyanka Gandhi
Assistant Professor
Department of Information Technology
Jagan Institute of Management & Sciences
talk2priyankap@gmail.com

Abstract

*The stronghold of the agile approach in the life cycle of the software development is the factor for the tremendous growth of the agile in the recent years.*It has grown as an flexible approach with greater requirement volatility which mainly stresses on good collaboration between developers and customers, and it support frequent and early delivery of the product. The idea of agile technology is "new" and research is in the early stage. *The paper takes tracks towards exploration of strengths and weaknesses of agile methodology in software's, based on which* ***SWOT****(Strength-Weakness-Opportunities-Threats) analysis is performed*No doubt the technology is adopted with greater enthusiasm by the developers yet the early and widespread adoption is the cause for the examination of its strengths and weaknesses. The analysis by the paper may act as a useful tool for highlighting and addressing the issues in the agile processes.

Introduction

Today's Information Technology (IT) manager is under ever-increasing pressure to deliver results – in the form of applications that drive improvements to the bottom line – even while IT budgets are being significantly slashed. Meanwhile, despite the fall of the Internet economy business environments continue to change at a rapid pace leaving many IT shops struggling to keep up with the pace of change. These changes have led to an increased interest in agile software development methodologies with their promise of rapid delivery and flexibility while maintaining quality.

There are several software development methodologies in use today. Some companies have their own customized methodology for developing their software but many methodologies are categorized as: heavyweight and lightweight.

Agile Software Development

Software development projects are highly unpredictable and that is why so many software development projects fail to meet expectations. The reason for why the Agile Software Development is needed for software development can be explained as below:

"If a predictive method like Waterfall is to avoid pain of problems, then we can say that Agile was developed to solve the pain of problems"

-M.Scott Peck, The Road Less Traveled

The Agile Software Development is a group of software process methodologies for small or medium organization is also called "agile". It isn't a set of tools or a single methodology. A recent survey conducted by Dr. Dobb's Journal shows 41 percent of development projects have now adopted agile methodology and agile techniques are being used on 65 percent of such projects. The idea of the Agile Development Framework is to create a pain free working environment for those small, collocated, self-organized teams in order to assist companies to take full advantage of the customer value of the delivered software product. On an Agile project, developers work closely with their customers for quick feedback. Therefore, the business contract will not become a barrier between customers and developers, but a platform to help customers and developers work together.

Agile is not the magic bullet many are claiming it to be. As many successes as there are in agile development, there are also many failures in both the adoption and use of Agile in software development.

So, Our Paper has been carried with the objective to analyze the benefits, weakness and opportunities. Most commonly used methods will examined from the angle of their applicability, strengths and weaknesses and their adoption in industry. In order to investigate and analyze, there is a need to compare the issues in the literature, research studies and industry. This will lead us to find benefits, limitations and difficulties in transition from traditional to agile software development.

Strengths

- **Requirements Inflation**
  The various features which are not identified at the beginning of the project creates a threaten estimates and timelines for the project progress in the traditional software development technology. But the Agile practices in software's are plan in the regular trade-off discussions about features and estimates at every iteration boundary. Changes and requirements inflation are accepted as a fact of software projects. Rather than utilizing change-suppression mechanisms, prioritization sessions are scheduled that allow worthwhile changes to proceed and initially envisioned features to be superseded if the business gives their approval. It has never been possible to squeeze a pint into a quart cup, but now at least we anticipate the likely issue and have mechanisms in place to address the matter as part of the project from its early stages.
- **Early benefits to the user/business**
  Agile methodology gives an early view to the final product might look and behave. This helps them into finalizing the user requirements**.**
- **Regaining the Productivity of development**
  When there is long project timelines, the sense of urgency to work in is often absent resulting to time lost in early project stages that can never be regained but the Agile methods recognize Parkinson's Law and the Student Syndrome apply to software projects. Parkinson's Law says that: "Work expands to fill the time available" and Student Syndrome: "Given a deadline, people tend to wait until the deadline is nearly here before starting work." By having short iterations, work is time-boxed into a manageable iteration and there is always a sense of urgency. Agile methods do not specifically address getting the right people on team, coaching and team development, but these are core leadership roles applicable to both agile and traditional projects.
- **Reduction of Employee "Bus-Factor"**
  The departing of the key personnel from the project team with significant information derails the project development in the conventional approach .Agile projects practice information sharing techniques such as common code ownership, pair programming, and frequent reporting at daily stand-ups etc specifically to decrease the "bus-factor" (the impact to the project of a key member being hit by a bus). When this "bus factor" is reduced multiple team members share key information and the hazard due to employee turnover is minute. Also, often overlooked, is the fact that when working in an engaging, rewarding, empowered, collaborative environment such as agile projects, people are far less likely to want to move elsewhere so the risk is often avoided as well as reduced.
- **Elimination of Inherent schedule flaws**
  Initially the Software development is inherently difficult to estimate and schedule. But on using agile technology for software projects, the team is heavily involved in planning and estimating through activities such as XP's planning game and Wideband Delphi workshops. By working in short increments the true velocity of the team quickly emerges and is visible to all stakeholders who are now more closely involved in the project. In short, the true progress is hard to hide and quickly revealed, giving feedback to the stakeholders.
- **Conflicting Requirements Breakdown**
  Agile projects play the role of product manager byutilizing the concept of subject matter expert, an ambassador user, or customer proxy. The idea is that group needs to be readily available to answer questions and make decisions on the project while the conventional projects suffer specification breakdown when no one will own the role and conflicting assumptions or decisions are made. Agile projects have some form of product owner role central to their core team composition to ensure decisions are made on time.

Hence agile methodologies have an adaptive team which is able to response to the changing requirements and does not have to invest time and efforts. The

documentation is crisp which also saves time. The face-to-face communication and continuous input from the customer leave no space for the guess work. The end result is the high quality software in least possible time duration with satisfied customers.

Weakness

When adopting the agile world with reckless abandon the developers particularly those at larger businesses should take a closer look to the technology. The technology suffers numerous weaknesses that are failed to acknowledge.

- **"It's Agile – not agile"**
  The true Agile is rarely practiced which is the biggest problem. In many cases, software developers, development managers and consultants alike mistake Agile for its lowercase sibling, agile, and assume that Agile is all about flexibility and absence of process.This is far from the truth.
  Agile has formal rules and structure, though they are quite a bit different from those of other development approaches. Agile is iterative, it is adaptive and it is supported by some outstanding tools and techniques. Most important, Agile is not anarchy. It does not mean that everyone does whatever they want and there's no sense of organization, despite the fact that you may feel this is the case.
- **Larger re-workduring combination of components**
  Due to the lack of long-term planning and the lightweight approach to architecture, re-work is often essential on agile projects when the various components of the software are combined and forced to interact. In a worst-case scenario, major portions of code may need to be re-written when two or more developers are not in sync, resulting in higher and higher re-work costs as the number of iterations increases.
- **Strong essentiality of Agile Manifesto Principle**
  Reflected in four principles and five of the Agile Manifesto,**Heavy Customer Interaction** is one of the biggest benefits of Agile, but it also becomes a weakness in some environments. One important thing to know about agile development teams is that they are high maintenance – they work quickly, but they also require much more time and attention from the business client side to be able to work quickly. If that time cannot or will not be spared for their benefit, using an agile approach will bring little gain over a Waterfall approach.
- **Agile is weak on architectural planning.**
  The software architecture is not entirely like construction architecture, but it does share certain qualities. Like in the construction of a building, architect determines the structure of the building, the materials to use and integrates the architecture into the landscape long before the first nail is hammered. Similarly, reliable software architecture and the software platform are selected before the heart of the software is developed. Otherwise, there's a greater chance of much work to be thrown away. The weak-architecture problem is only an issue when the architecture and the platform are not known or pre-determined. In many larger businesses, software architecture and platform selection are done separately from software development via enterprise architecture and architectural roadmaps, which often takes the burden away from the agile software team. However, if the platform is not known prior to the start of the project and the architectural approach is new or unproven, then the agile software approach will resist defining and dealing with architecture.
- **Difficulty in coordination for large projects and large organizations.**
  With its strong emphasis on customer interaction, self-organizing teams, verbal-communication over-written-documentation, prototyping and requirements flexibility, it becomes extremely complicated to coordinate work as an agile team grows. The scalability problem with Agile is not minor and it shouldn't be overlooked.
- **Agile thrives with co-located teams.**
  In a typical agile environment, the agile development team and the business users are located in the same physical location, such as the same floor and cube space, to increase interaction within the team and with the customer. This technique is highly effective, but it is also not always practical. In many cases, the

line-of-business does not have the space to temporarily house the development team members.

- **Little emphasis on core aspect of Project Management**
  Agile has limited project planning, estimating and trackingthrough the use of backlogs such as prioritized to-do lists of software product features by fixing deadlines instead of scope. So in some cases, the lightweight planning, estimating and tracking process suits small, non-strategic projects well.

Opportunities

The agile development methodology provides many opportunities to assess the direction of the project throughout the development life cycle but the major opportunities rely under the following headings:

- **Iterative and Incremental methodology**
  In waterfall the teams only have one chance to get each aspect of a project right but in an agile every aspect of development whether it is requirement of design is continually revisited through out the life cycle. When a team stops and re-evaluates the direction of a project in every two week there is always a time to move in other direction .thus by focusing such repletion of work cycles the agile methodology is described as iterative of incremental approach.

- **Inspect and Adapt approach**
  This approach is the result of iterative technique of agile development due to which the developers greatly reduce both the development time and cost to the market because the teams can gather requirements at the same time they are developing the application. The team work cycles are also limited to two weeks which provides the customers with opportunities to calibrate the success in the real world.

- **Rapid Action Development**
  The processes such as agile and scrum can both enhance or threaten the user experience quality by overcoming the usability barriers in the traditional development.

Threats

- **Necessary documentation with lack of structure**
  Various agile methods are greatly prescriptive of important practices and have structure which is informal.

- **Requires greater cultural change to adopt**
  The agile successes can be very inefficient if the requirements of one area of code change through various iterations, the same programming may need to be done several times over whereas if a plan were there to be followed, a single area of code is expected to be written once.

- **Tool to understand project velocity**
  Impossible to develop realistic estimate of work effort needed to provide a code because at the beginning of the project no one known the entire requirement. Most of the agile methods provide a better tool to understand the project working. The agile methods by putting the software in customer hands give early discovery of the requirements and upfront planning of the project.

Conclusion

In this paper we have provide a detail analysis of agile approach with its core strengths, weaknesses and threats to the real world. Furthermore, my conclusion is that the agile development framework is recommendable for software development processes nevertheless it is not suitable for all development projects.

Agile is more than just a set of process steps and practices. Agile surely has some fundamental practices based upon the model that is selected for implementation but in addition agile gives us core

manifesto, principles and values. This is what we feel sets it apart from the rest. We all know that emotional intelligence and treating people like assets rather than pieces on a chessboard makes common sense but Agile puts this into the forefront of everything. Westrongly believe that agile focus on beliefs, principles, values, and the way that it empowers people through servant-leadership, which results in agile teams feeling more **engaged**. It is this engagement, passion, and focus on more than just processes and practices that sets Agile apart, that makes us fight this war so hard

In this paper we have tried to build a road map that can led organization towards their target of agile adoption. There are many routes an enterprise can take, but, hopefully, some of the impediments we have identified will help a company to make it there. One thing is clear that there is no "*one-size-fits-all*" solution for all development projects.

References

1. T. Dyba. Improvisation in small software organizations, IEEE Software 17 (5) (2000) 82–87
2. Jennings N. R. Building Complex Software Systems: The Case for an Agent-based Approach. *Communications of the ACM*, Forthcoming, 2001.
3. Labrou Y., Finin T. and Peng Y. Agent Communication Languages: The Current Landscape. *IEEE Intelligent Systems*, 14(2), March/April 1999 1999.
4. P. A Gerfalk, B. Fitzgerald. Flexible and distributed software processes: old petunias in new bowls? Communications of the ACM 49 (10) (2006) 27–34
5. G. Keefer, Extreme Programming Considered Harmful for Reliable Software Development 2.0, AVOCA GmbH, Online Report, 2003.
6. Schwaber K. and Beedle M. Agile Software Development with Scrum. Prentice Hall, 2001.
7. **M**iller, G., "Want a better software development process'? Complement it," IT Professional, vol.5, no.5, pp. 49-51, Sept.-Oct. 2003.
8. M. Poppendieck, T. Poppendieck, Lean Software Development – An Agile Toolkit for Software Development Managers, Addison-Wesley, Boston, 2003, ISBN 0-321-15078-3.
9. P. Meso, R. Jain, Agile software development: adaptive systems principles and best practices, Information Systems Management 23 (3) (2006) 19–30.
10. S.Nerur, V. Balijepally, Theoretical reflections on agile development methodologies, Communications of the ACM 50 (3) (2007) 79–83.
11. D. Cohen, M. Lindvall, P. Costa, An introduction to agile methods, in: M.V. Zelkowitz (Ed.), Advances in Computers, Advances in Software Engineering, vol. 62, Elsevier, Amsterdam, 2004.