IJCSE International Journal of Computer Science and Engineering Open Access
Review Paper Volume-2, Issue-3 E-ISSN: 2347-2693

Study: Impact of Agile on Current IT Scenario

Anuradha Singhal[1*], Tarun Chaner[2] and Richa Gupta[3]

[1*]*Department of Computer Science ,Delhi University,India, anuradhasinghal19@gmail.com*
[2] *Department of Computer Science, Kurukshetra University,India,tarun.chander@gmail.com*
[3]*Department of Computer Science , Delhi University India ,India ,richie.akka@gmail.com*

www.ijcaonline.org

Received: 2 March 2014 Revised: 10 March 2014 Accepted: 22 March 2014 Published: 30 March 2014

Abstract— The present research paper majorly discusses with regard to various issues related to agile software development approach in related to changes mande in diverse background.

Index Term—Extreme Programming, Feature driven development, Focal Point, Return on Investment (ROI), Scrums

I. INTRODUCTION

Agile software development is an evolutionary, highly collaborative, disciplined, quality-focused approach to software development, whereby potentially shippable working software is produced at regular intervals for review and course correction. Agile software development processes[1] include Scrum, Extreme Programming (XP), Open Unified Process (OpenUP), and Agile Modeling (AM), to name a few. Agile development is becoming widespread because it works well – organizations are finding that agile project teams, when compared to traditional project teams, enjoy higher success rates, deliver higher quality, have greater levels of stakeholder satisfaction, provide better return on investment (ROI), and deliver systems to market sooner. But, just because the average agile team is more successful than the average traditional team, that doesn't mean that all agile teams are successful nor does it mean that all organizations are achieving the potential benefits of agile to the same extent. Agile processes are intended to support early and quick production of working code. This is accomplished by structuring the development process into iterations, where an iteration focuses on delivering working code and other artifacts that provide value to the customer and, secondarily, to the project. Agile process proponents and critics often emphasize the code focus of these processes. Proponents often argue that code is the only deliverable that matters, and marginalize the role of analysis and design models and documentation in software creation and evolution. Agile process critics point out that the emphasis on code can lead to corporate memory loss because there is little emphasis on producing good documentation and models to support software creation and evolution of large, complex systems. The claims made by agile process proponents and critics lead to questions about what practices, techniques, and infrastructures are suitable for software development in today's rapidly changing development environments. In particular, answers to questions related to the suitability of agile processes to particular application domains and development environments are often based on anecdotal accounts of experiences.

Corresponding Author: *Anuradha Singhal*

II. LITERATURE REVIEW

There is a demand for approaches able to deal with the increasing complexity of software development, because coordination becomes more difficult when complexity increases (Kraut, 1995)[8]. A family of potential approaches that has received a lot of attention from software engineers and software researchers the later years has adopted the term "agile" (Abrahamsson, 2003)[1]. Agile software development is introduced as a software development approach promoting teamwork, innovation, flexibility, and communication (Agerfalk, 2006)[2]. Agile development approaches and global software development approaches differ significantly in their key tenets, e.g. regarding coordination mechanisms (Ramesh, 2006)[9]. Global software development focuses on command-and control and formal communication. The desired organizational structure is mechanistic, which means that it is bureaucratic with high formalization. Agile or change-driven development focuses on leadership-and-collaboration and informal communication. The desired organizational form is organic, which means that it is flexible, participative, and encourages cooperative social action. Therefore, applying agile principles to global software development marks an intersection of two seemingly incompatible approaches. Still, (Ramesh, 2006) demonstrate how a balance between agile and distributed approaches can help meet the challenges with incorporation of agility in distributed software development. Despite the differences, there is a growing interest in assessing the viability of using agile practices for distributed teams (Agerfalk, 2006)[2]. Several reports claim that it can be done successfully ((Berczuk, 2007)[3], (Farmer, 2004), (Fowler, 2003), (Holmstrom, 2006)[6], (Nisar, 2004), (Korkala, 2007)[7], (Ramesh, 2006), (Sulfaro, 2007), (Sutherland, 2007))[10].Agile software development comprises a number of practices and methods ((Erickson, 2005)[5], (Cohen, 2004), (Abrahamsson, 2002). Among the most known and adopted agile methods are Extreme Programming (XP) (Beck, 2004) and Scrum (Schwaber,

2001). XP focuses primarily on the implementation of software, while Scrum focuses on agile project management (Abrahamsson, 2003). In this study the focus is on Scrum since Scrum is an agile approach to the management of software development projects ((Erickson, 2005), (Cohen, 2004)[4], (Abrahamsson, 2002)), and thus focuses on the coordination of work.

III. RESEARCH METHODOLOGY

Methodology is defined as a process in which particular discipline is followed by the system of methods. Research in common refers to a search for knowledge. Research can also be defined as a scientific and systematic search for relevant information on a particular topic. The research is in fact a scientific investigation. The main purpose of the research is not only to discover the answers to the questions specified but also to provide certain scientific procedures throughout the application. Thus, research methodology is generally an association of considerable part of most important scientific research and technology. There are many features which are obtained in the past based on the qualitative and quantitative approaches towards the researchers.

Quantitative vs. Qualitative: Quantitative research depends on assessment of quantity or amount i.e. the organization is based on measurement of total quality or amount details. The details are applicable based on the phenomena which are expressed as per the terms of quantity. It is relevant to the outcomes that can be expressed corresponding to quantity. Qualitative research is concerned to qualitative phenomenon, i.e., it is a phenomenon that involves quality or kind.To address the research questions, semi-structured interviews were conducted with the persons. The focus was on understanding coordination of work, communication within and between the teams, feedback-sessions, planning and estimation, use of documentation, roles and specializations, and how decisions were made. All the interviews were transcribed. The interviewees were asked to indicate the relative level of the various coordinating mechanisms used between local and remote sites in their projects. While the interviews were the primary source of data for this study, access was given to previously collected data on the primary company. The research methodology for this report entails primary and secondary sources available and is of qualitative approach.

IV. SAMPLING

4.1 *Agile Adoption Survey*
February 2008
Message sent out to DDJ mailing list
642 respondents:
54.8% were developers,
29.4% were in management
41.6% had 10-20 years IT experience,
37.2% had 20+ years
37.7% worked in orgs of 1000+ people
71% worked in North America,
17% in Europe,
4.5% in Asia

4.2 Project Success Survey
August 2007
Email to DDJ mailing list
586 respondents
54% were developers/modelers,
30% were in management
73% had 10+ years in IT
13% worked in orgs of 1000+ IT people
84% worked in commercial firms
69% North American,
18% European

V. ANALYSIS

5.1 Case study of IBM

The following case study was described at the Agile 2008 conference by Sue McKinney, Vice President IBM Software Group, and Ted Rivera, Senior Agile Transformation Architect. International Business Machines is a large $100B worldwide technology and services company. Ten years ago like most companies, software development at IBM was done using the old-fashioned waterfall approach. This resulted in significant quality issues and ever-increasing schedule slips. After several disappointing years, IBM shifted to the Rational Unified Process (RUP) approximately 5 years ago. The iterative nature of RUP seemed to help initially, but the development teams still struggled to meet schedules with high quality deliverables. In 2006, IBM invested in an elaborate shift over to Agile Methods. The shift involved over 25,000 software developers across 78 worldwide locations. 220 commercial projects started up with a commitment to following the philosophies and principles of Agile Methods. Measurements were taken using the IBM deployment evaluation framework, tracking numbers of lines of code, employee satisfaction surveys, and capturing defect trends. Overall, the projects experienced a 30% velocity increase when compared to previous non-agile projects. This resulted in substantial measurable gains in time to market and ROI. One of these 220 projects dealing with the IBM WebSphere product was enormously successful and met its yearly sales goal on the very first day of availability. In addition, $2.567M was saved by increased software quality and earlier defect detection. Field defects were reduced by 80% from previous releases. The number of trouble tickets from the field customers was reduced by 95% from previous releases. Another of the 220 projects involved the IBM DB2 product and 600 software engineers. Comparing previous non-agile releases of DB2, IBM found that using Agile Methods led to a 25% productivity gain and a saving of over $5M per year.IBM is an excellent example of a large company that has fully embraced Agile Methods and is currently realizing the fruits of this decision with substantial measurable cost avoidance and increased sales

revenue.IBM has since long recognized the value of lean and agile development. IBM had been applying rapid application development, lean approaches, model driven development and many other agile principles regularly, to ensure quality deliverable at a much quicker pace. With more well defined agile methodologies like Scrum and XP becoming popular, IBM has been leveraging these successfully, delivering projects using one or more of these agile methodologies, either in their out-of-the-box form, or using a modified version. IBM Agile projects collaborate using our internal knowledge management systems, communities of practice, and have access to toolsets and enablers hosted in the centralized organizational process repository .

The following are the responses to corresponding questions that are acquired from respondents.

Fig 1 shows adoption of Agile Technology in industry.

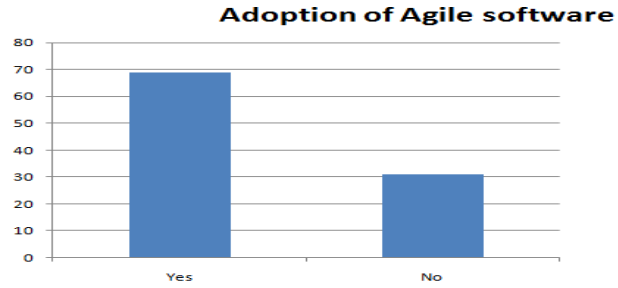

Fig 1 Adoption of Agile Software

18.3% of respondents indicated they're still in the pilot stage.15% of "No" respondents hope to do Agile this year.

Fig 2 depicts Impact of Agile on Technology. Fig 3 shows Impact of Agile on Sales . Fig 4 illustrates Impact of Agile on Business StakeHolder satisfaction. Fig 5 illustrates Agile impact on system development cost

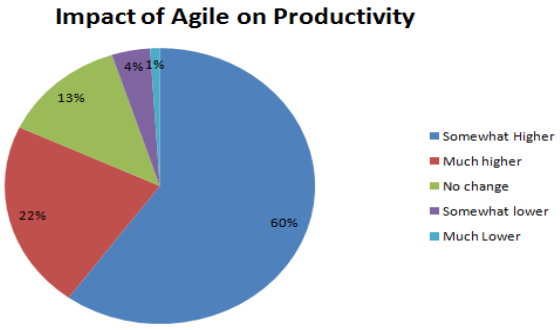

Fig 2 Impact of Agile on Productivity

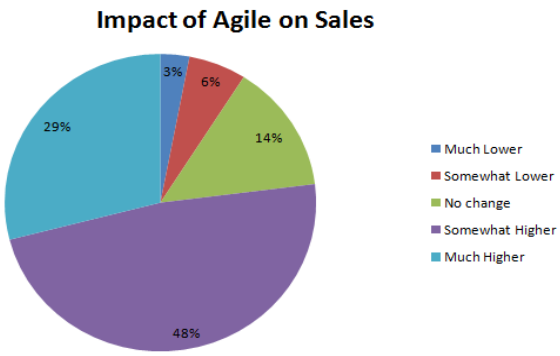

Fig 3 Impact of Agile on Sales

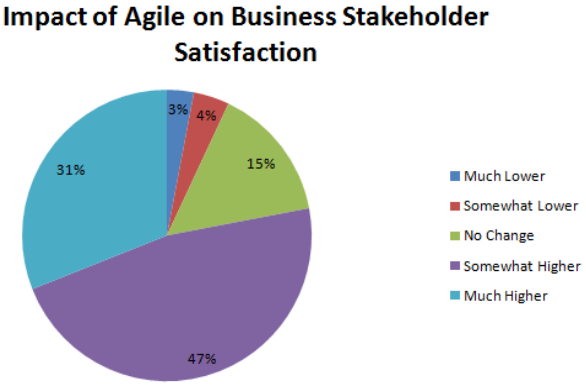

Fig 4 Impact of Agile on Satisfaction of Stakeholders

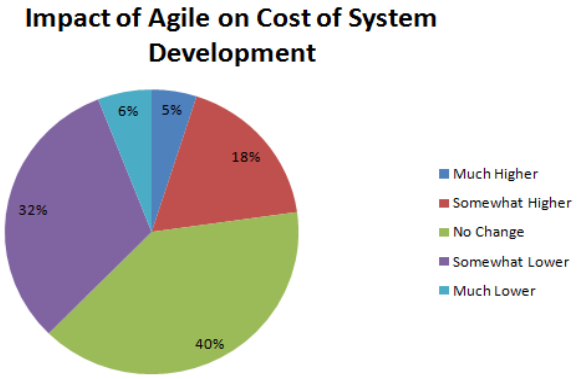

Fig 5 Impact of Agile on cost of System Development

Fig 6 shows Adoption of various techniques and impact on productivity.

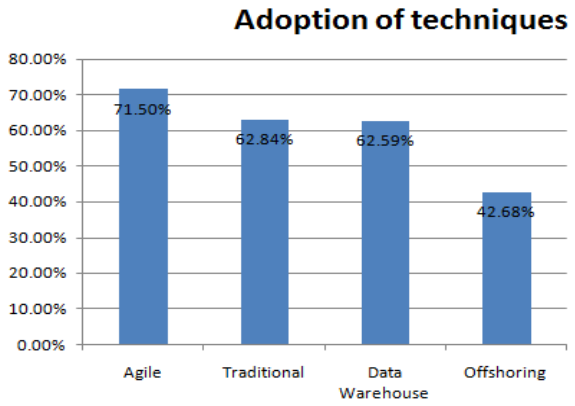

Fig 6 Adoption of various techniques

Quality: 87.3% believe that delivering high quality is more important than delivering on time and on budget

Scope: 87.3% believe that meeting actual needs of stakeholders is more important than building the system to specification

Money: 79.6% believe that providing the best ROI is more important than delivering under budget

Staff: 75.8% believe that having a healthy workplace is more important than delivering on time and on budget

Schedule: 61.3% believe that delivering when the system is ready to be shipped is more important than delivering on schedule

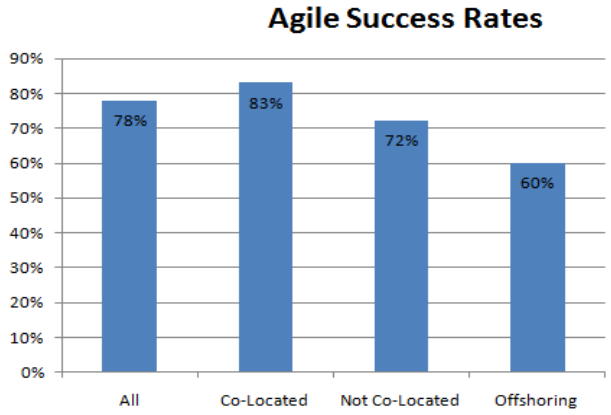

Fig 7 Agile methodology success rates

Agile methodology success rates is depicted in Fig7. (214 co-located projects, 210 not co-located, and 129 offshoring /outsourcing). Percentage of Agile Technology knowledge acquisition is shown in Fig 8.Fig 9 ,10depicts types of Agile methods and Agile techniques used in today IT world.

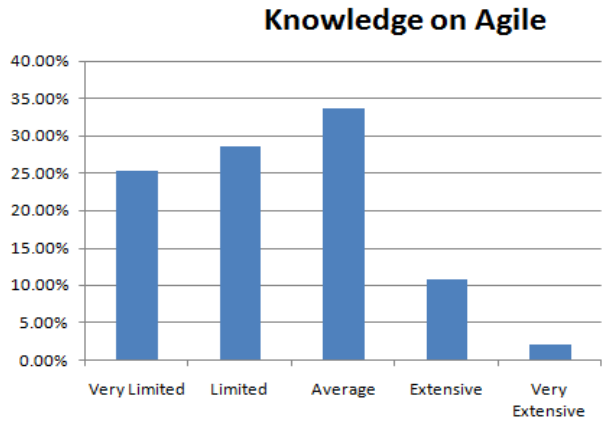

Fig 8 Knowledge on Agile

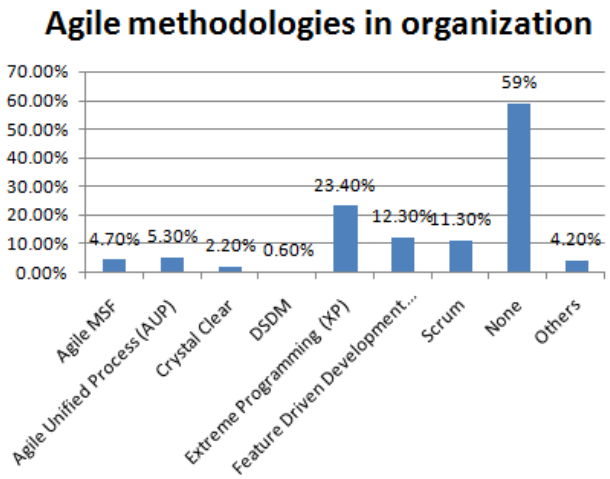

Fig 9 Agile Methods used

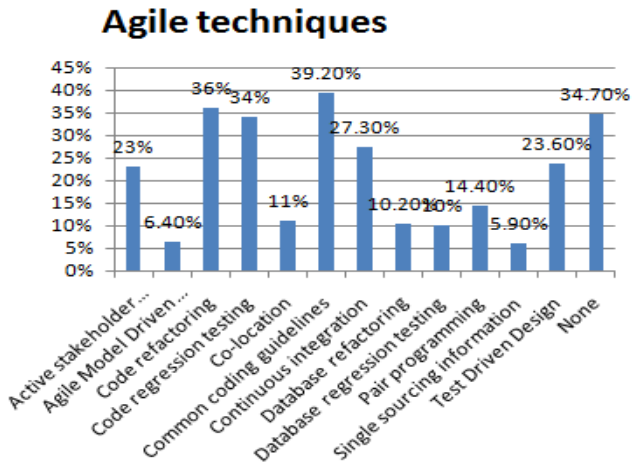

Fig 10 Agile Techniques used

VI. LIMITATIONS OF AGILE PROCESSES

1.1 Limited support for distributed development environments-

The emphasis on co-location in practices advocated by agile processes does not fit well with the drive by some industries to realize globally distributed software development environments. Development environments in which team members and customers are physically distributed may not be able to accommodate the face-to face communication advocated by agile processes.

1.2 Limited support for development involving large teams

Agile processes support process "management-in-the small" in that the coordination, control, and communication mechanisms used are applicable to small to medium sized teams. With larger teams, the number of communication lines that have to be maintained can reduce the effectiveness of practices such as informal face-to-face communications and review meetings. Large teams require less agile approaches to tackle issues particular to "management-in-the-large".

Traditional software engineering practices that emphasize documentation, change control and architecture-centric development are more applicable here.

1.3 Limited support for developing large complex software

The assumption that code refactoring removes the need to design for change may not hold for large complex systems in particular. In such software, there may be critical architectural aspects that are difficult to change because of the critical role they play in the core services offered by the system. In such cases, the cost of changing these aspects can be very high and therefore it pays to make extra efforts to anticipate such changes early. The reliance on code refactoring could also be problematic for such systems. The complexity and size of such software may make strict code refactoring costly and error-prone.

Agile development methods aren't used under the following circumstances:

i) When goal is to produce documentation, such as a requirements document, for sign-off by one or more project stakeholders
ii)While using a case tool to specify the architecture and/or design of our software BUT not using that specification to generate part or all of our software
iii)When customers/users have limited involvement with our efforts. For example they are involved with initial development of requirements, perhaps are available on a limited basis to answer questions, and at a later date will be involved in one or more acceptance reviews of the work.

VII. CONCLUSION

The agile qualities in large IT shops have always sounded philosophical. This perception is changing. Recent facts and studies support the notion that large IT shops today can quickly deliver large & efficient software using Agile methodology. Therefore, the better question is how a large unit can act like small in agility, but remains big in risk management. When we tend to emphasize on the latter, we move towards the classical Waterfall approach, which has hidden cost in terms of requirements gaps and probable program failure. We then try to be agile by retaining most of the heavy practices but making the software development lifecycle iterative or spiral. In other words, we simply loosen the controls and start calling it agile development, and expect it to behave like one as well. Many a times, we are not conscious of the best practices that Agile methodologies use to reduce the risk of eliminating controls. For instance, Scrum has the principle of "documenting less but broadcasting more" - a mechanism that makes the project inherently risk-aware despite having less artifacts.

Today, Agile is at crossroads not in terms of "how loose" or "tight" should the software development lifecycle behavior be, but in terms of how it brings in the right collection of best practices, which would make the project both requirement-aware and risk-aware, ensuring high flexibility, reduced risk and increased visibility. This correction in our perception of Agile has happened rather late – I claim so as I see an explosion in the adoption of Agile methodologies that exploit this confusion. What is needed is the practices that the IT shop would institutionalize to hedge the absence of many merits that traditional waterfall still carries. This diminishes role of methodologies and heightens the need for a best practice culture.

REFERENCES

[1]. Abrahamsson, P. W. J. S. M. T. &. R. J., **2003**. New directions on agile methods - A comparative analysis. International Conference on Software Engineering-ICSE, pp. **244-254.**

[2]. Agerfalk, P. J. &. F. B., **2006.** Flexible and distributed software processes: Old petunias in new bowls?. Communications of the ACM, Volume **49**, pp. **26-34**.

[3]. Berczuk, S., **2007**. Back to basics: The role of agile principles in success with a distributed Scrum team.. In: AGILE 2007. s.l.:s.n., pp. **382-388**.

[4]. Cohen, D. L. M. &. C. P., **2004**. An introduction to agile methods.. Zelkowitz, M. V. (Ed.) ed. Amsterdam: Elsevier.

[5]. Erickson, J. L. K. &. S. K., **2005.** Agile modeling, agile software development, and extreme programming: The state of research. Journal of Database Management, Volume 16, pp. **88-100**.

[6]. Holmstrom, H. F. B. A. P. J. &. C. E. O., **2006**. Agile practices reduce distance in global software development. Information Systems Management, Volume **23**, pp. **7-18**.

[7]. Korkala, M. &. A. P., **2007**. Communication in distributed agile development: A case study.. 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, pp. **203-210**.

[8]. Kraut, R. E. &. S. L. A., **1995**. Coordination in software-development. Communications of the ACM, Volume 38, pp. 69-81.

[9]. Ramesh, B. C. L. M. K. &. X. P., **2006**. Can distributed software development be agile?. Communications of the ACM, Volume **49**, pp. **41-46**.

[10]. Sutherland, J. V. A. B. J. &. P. N., **2007**. Distributed Scrum: Agile project management with outsourced development teams. Hawaii International Conference on System Sciences , p**. 274**.

AUTHORS PROFILE

Anuradha Singhal ,MS(Software Sytems) , BITS Pilani B.E Rajasthan University Tarun Chander , B.E (Computer Science) Kurukshetra University Richa Gupta .MSc(Computer Science) ,Delhi University