Status of User Requirement Engineering in Agile and Non-agile Software Development

1Jaskirat Kaur
1Assistant Professor
1sri guru granth sahib world university, fatehgarh sahib, punjab

Abstract - **Over recent decades, there is a steady change in rate of technology, businesses types, market pressures, competitive threats that directly challenge the style of developing software. Innumerable software development methods exist and are typically classified into traditional and agile methods. Eliciting, analysing, documenting and managing variation in user requirements are the most prime and pivotal segments of software development life cycle. Because inadequate requirements can lead to budgets overruns, schedule overruns poor quality systems or system failures. This paper explores the current requirement engineering practices pursued by agile and non-agile software development teams. We also analyse which software development methodology is in tread and leading to fruitful results. The aim is to acknowledge the project managers and software developers about the revolution in development area so they can adopt the best principles to serve requirement engineering activity.**

keywords - **Requirements Engineering, Requirements Analysis, Requirements Elicitation, Agile Software Development, Traditional approach**

I. INTRODUCTION

Standish Group's CHAOS report in 1991 emphasis that 48.1% of the proportion of projects failure is caused by requirement affairs. In spite of numerous technical innovations and research work, projects still fail or cannot meet key functions mainly due to insufficient operational requirements. So eliciting requirements effectively from those who need a software-intensive system to the software developing team is a very imperative task in SDLC due to various issues:

- Clients don't have a fair picture of their requirements in the beginning and often demand new requirements even after the cost and schedule have been fixed by denying to a set of pre-specified requirements.
- Due to poor communication between clients and developers, clients don't participate in review meetings.
- Technical personnel use jargon unconsciously while clients are technically weak and wonder how to express their needs.

Then analysing, managing the gathered and varying requirements of users wants further extra attention because of following reasons:

- Developers start coding instantaneously before they actually understand the whole requirement from analyst, which causes lots of defect fixing or reworking in test/verification phase.
- Technical personnel and clients may have different vocabularies and may wrongly believe that they are in perfect agreement.
- Developers do not develop a system specific to the client needs rather try to make the requirements fit an existing system rather than
- Programmers often perform analysis themselves instead of engaging personnel with the domain knowledge to understand a client's needs properly.

Thus eliciting and analyzing user requirements are undoubtedly vital step to achieve effectiveness in management and implementation of any product.

II. TRADITIONAL REQUIREMENT ENGINEERING VS AGILE REQUIREMENT ENGINEERING

In traditional way of requirement engineering, software development teams serve requirements that tend to transform very quickly and become antiquated even before project completion. Fast development in industry and technology seriously challenge the software development based on pre-specified requirements and to alter the traditional environment of requirement engineering. Hence agile requirements engineering emerges. Agile approach consists of a set of principles that endorse close collaboration between customer and software development teams, as against to development only by software teams; welcome changing requirements by customers even late in development, as against to defining a fixed set of requirements at the beginning; frequent delivery of portions of working system, as against to final delivery of the complete product at the end; and adaptive organizational capability of teams according to changing business requirements. Agile requirements engineering is distinct from traditional way of requirements engineering based on different phases of Requirement Engineering in the following manner:

- Requirement Capture:

In traditional requirements engineering there is a communication gap between clients and developers and the interest of client groups is inconsistent and difficult to reconcile. While in agile requirement engineering clients and developers communicate more frequently and closely i.e. there is face to face communication between them and clients are usually available on-site. Ideal client representatives are engaged to reconcile conflicts of interest.

- Requirement Modeling and Prioritizing:
  In traditional requirements engineering, requirement priority is only taken when project first started and requirement modeling would be saved as part of requirement document. But in agile approach requirement priority is checked at each stage of development cycle and agile modeling is "throw-to-use" that only helps to understand the software.
- Requirement Specifications:
  In traditional requirements engineering, massive requirement documentation is developed with complete user requirements when project first started and is not updated frequently. It is suitable for large teams, for explaining the same thing to pretty different people is really unworthy. There is low level of clients' participation. While in agile methods vague modest-sized requirement documentation is developed that confirms specific requirements before the next development cycle and is updated very frequently. It is suitable for small teams, as communication could run more smoothly in them. There is high level of clients' participation.
- Requirement Verification:
  In agile requirement engineering, requirements are verified by conducting inspective meetings with clients and reception testing is done. Inspective meetings confirm the direction of projects, gain trust between clients and teams. Thus, agile development throws emphasis on "testing first". While in traditional approach, no such meetings are conducted but here prototyping emphasize developing gradually and getting feedback quickly.

III. Analysis and Results

The objective of the present research is to study the existing requirements engineering practices being carried out in the different software companies and also to identify and analyse which requirement engineering methodology leads to better results in software projects.

First twelve independent variables were recognized from literature review and agile manifesto i.e. most of them are optimized set of agile principles. In our study we relate each of these 12 variables with another variable called Success. Success include five things- Reduced delivery schedules, increased return on investment, increased ability to meet with the current customer requirements, increased flexibility to meet with the changing customer requirements, improved business processes. Survey questions were formulated from these 12 independent and one dependent variable. Table I lists all variables with their corresponding labels.

TABLE I
VARIABLES

Variable	Label
Organizational culture	Ind1
Team capability	Ind2
Personal characteristics	Ind3
Team distribution	Ind4
Communication and negotiation	Ind5
Customer satisfaction	Ind6
Customer commitment	Ind7
Customer collaboration	Ind8
Prioritization	Ind9
Team Size	Ind10
Reviews and Tests	Ind11
Planning	Ind12
Success (Dependent)	Ind13

Survey questionnaire consists of three sections. Section A collects general information about organization like company's primary business, company size, team size etc. Section B identifies the current situation and methodologies being adopted for requirement engineering. Section C identifies future lines for the research. Most of the survey questions are closed ended and the respondents were asked to rate each variable of requirement engineering practice on 7-point Likert scale ranging from "Strongly Agree" to "Strongly Disagree".

The collected data is statistical analyzed. First the reliability test was performed that reveals how consistent the questionnaire is and how consistently the survey respondents responded to the items. It was calculated with the help of statistical tool SPSS. A Cronbach's alpha with value greater than 0.6 is considered standard in survey research. Table II depicts reliability statistics.

TABLE II
RELIABILITY STATISTICS

Coefficient of reliability	Value of Cronbach's Alpha
Cronbach's Alpha	.742 (Acceptable)

According to survey results the majority of the respondents (around 75%) belong to organizations whose primary line of business is Software Development, Web Development and Testing. There were around 11% responses from organizations whose the primary line of business is networking, around 5% responses from organizations whose the primary line of business as banking/ insurance industries, around 7% each from medical/ health organizations, and around 2% organization have the primary line of business as education. There were no responses from Aerospace organizations.

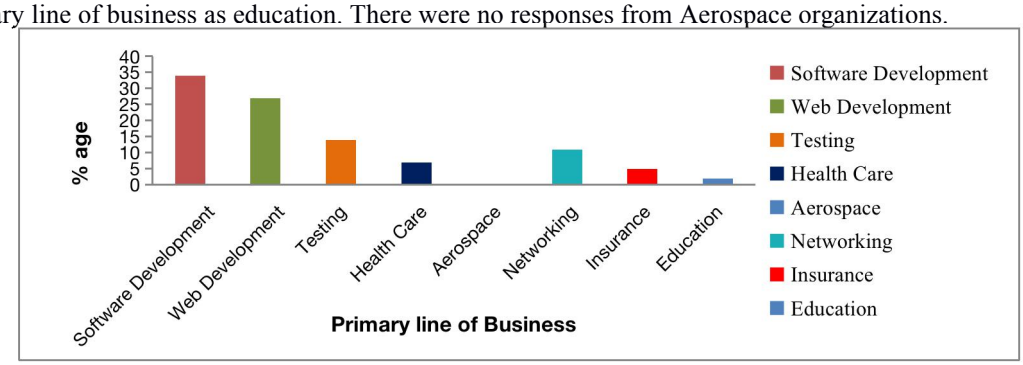

Fig1. Primary line of business of organizations

Of all the respondents, roughly 37% belong to organizations having 500–1000 employees, roughly 28% having more than 1000 employees, roughly 19% having 50–500 employees and roughly 16% having less than 50 employees. Thus, we can see that the majority of the respondents belong to organizations having more than 500 employees. So the number of responses from small-sized organizations is roughly around 35% and from medium-sized organizations is 37% and rest 28% is from large-sized organizations. Thus there has been fairly good response from both relatively small-sized and large-sized organizations.

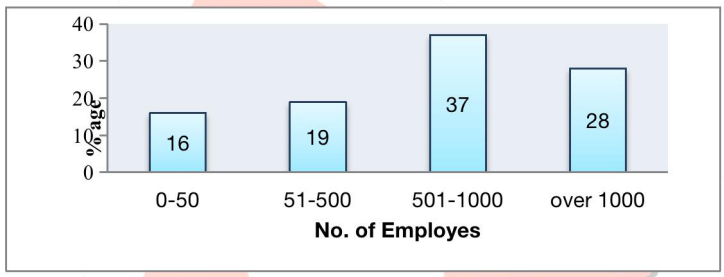

Fig 2. Number of employees in organization

Most of the responses (roughly 36%) came from respondents belonging to teams having strength of 10–19 employees, roughly 29% in teams having less than 10 employees. Around 12% responses each were obtained from teams having 20-29 employees, roughly 10% in teams having 30-39 employees, roughly 8% in teams having 40-49 employees and roughly 5% in teams having 40-49 employees. This shows a fair distribution of participation from all sizes of teams.

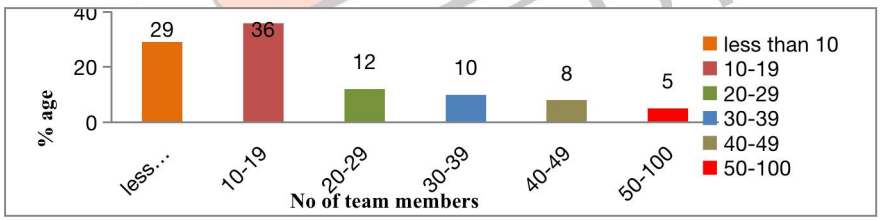

Fig 3. Number of team members in a project

The majority of the respondents (roughly 36%) have working experience for 1–4 years, 25% for 4–6 years, 20% for greater than 6 years, and 19% for less than 1year. This shows that we had participants from fairly all duration levels.

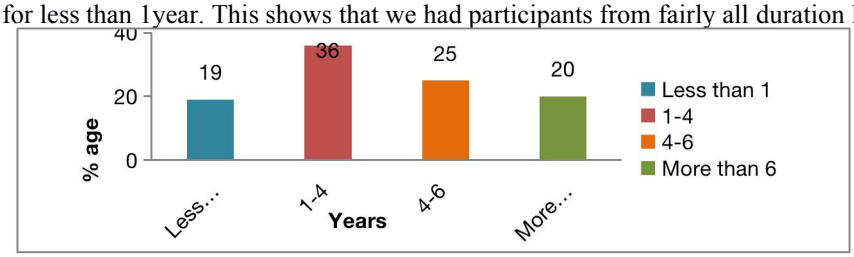

Fig 4. Working experiences of employees

In organizations most of the respondents (roughly 49%) use Scrum as agile method, followed by respondents who use Feature Driven Development (around 44%) and roughly 7% respondents use other agile methods.

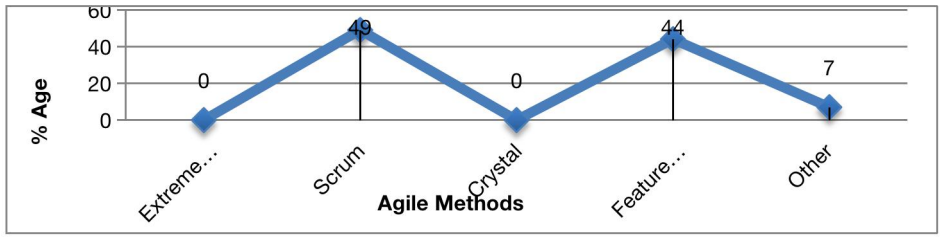

Fig 5. Agile methods used in organizations

Then we perform descriptive statistics to abstract the information latent in the collected survey data by calculating mean, and standard deviation. Table II shows the descriptive statistical results.

TABLE II

DESCRIPTIVE STATISTICS

Variables	N	Mean	Std. Deviation
Ind1	50	1.5760	.38469
Ind2	50	1.6800	.62073
Ind3	50	1.9320	.55345
Ind4	50	2.2500	.63286
Ind5	50	1.9400	.76692
Ind6	50	1.4000	.53452
Ind7	50	1.2800	.45356
Ind8	50	1.3800	.56749
Ind9	50	1.6420	.41458
Ind10	50	2.4674	.86527
Ind11	50	2.0440	.40515
Ind12	50	1.3200	.51270
Ind13	50	1.9720	.43894

By considering the mean values, amongst all the 12 independent variables, Ind7 (customer commitment) has the lowest mean of 1.2800 with a standard deviation of .45356 i.e. most of the respondents in the survey strongly agree and give highest priority to customer commitment more than anything else. Ind12 (planning), Ind8 (customer collaboration), Ind6 (customer satisfaction), Ind1 (organization culture), Ind2 (Team capability), Ind9 (prioritization), Ind3 (personal characteristics) and Ind5 (communication and negotiation) according to the survey results were also considered to be in the range of "agree, strongly agree". Ind10 and Ind 4, has the highest mean value, this shows that most of the respondents are "somewhat agree" with the statement that "whether most of the team members in their projects were geographically closely located, and the other teams that they interact with are geographically closely located" and with the statement that "personals work in small teams". It should also be observed that none of the independent variables has a mean score above 3 (in the 7-point Likert scale). This shows that on an average, the respondents that were surveyed were not in disagreement with the variables in their projects. This reveals that by adopting the all variables expect "team distribution" and "team size" among 12 independent variables the Success rate will increase in terms of return on investment, delivery schedules, to meet with the current and changing customer requirements. And there will be improvement in business processes

In order to assess which software development methodology among agile and traditional approach, is in tread and leading to fruitful results, the respondents were asked few more questions and the study reveals many more things like:

Majority of organizations i.e. about 51% support all agile approaches in fetching user requirements. Fig 6 depicts this picture.

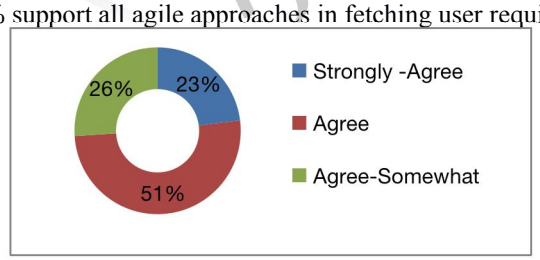

Fig 6. Organizations supporting all agile approaches

About 76% of the respondents agree the transition from non-customer-centric to customer-centric development should be adopted by all IT companies. Fig 7 reveals that the traditional way of requirements engineering is being replaced by agile approach at large extent.

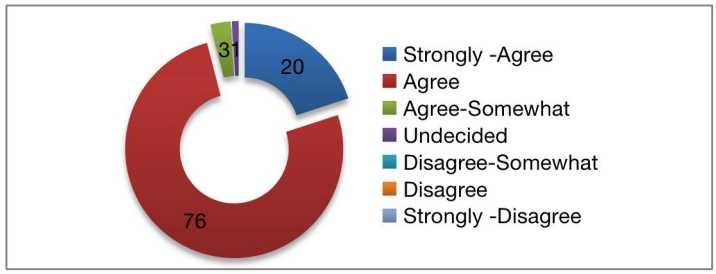

Fig 7. Transition should be adopted by all IT companies

Fig 8 says that about 60% of the respondents think quality assurance is easy and consumes fewer resources during projects in agile development as compared to traditional user requirements gathering approaches

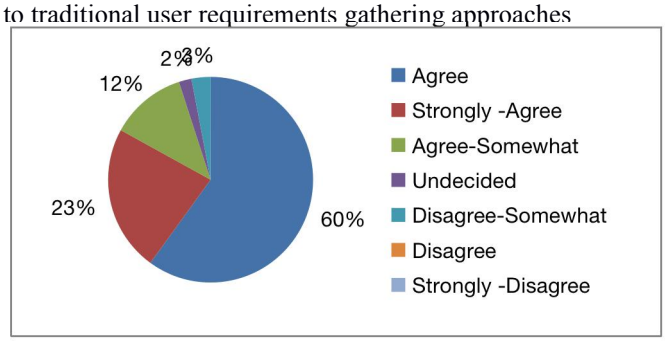

Fig 8. Agile approaches are better as compared to traditional approaches

About 60% of the total respondents think that the success rate will rise with the adoption of agile terms as compared to traditional user requirements gathering approaches and fig 9 depicts this state of revolution.

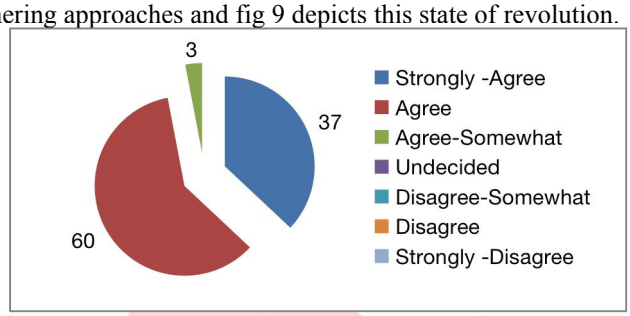

Fig 9. Success rate will rise with adoption of agile terms

IV. Conclusion

Requirement engineering is the core activity in the software development life cycle and if requirements are not clear, it will act as one of the main cause of project failure. Survey study reveals that agile requirement engineering principles provide more customer satisfaction, minimize bug rates, shorten the delivery schedules, increase in return on investment, and accommodate changing customer requirements during the development process. Hence traditional way of requirements engineering is being replaced by agile approach at large extent.

References

[1] The Standish Group, The CHAOS Report, "Computing Degrees & Careers", [Online] Available: http://www.projectsmart.co.uk/docs/chaos-report.pdf.

[2] A. Sillitti and G. Succi, *Requirements Engineering for Agile Methods*, In Engineering and Managing Software Requirements, pp. 309-326, Springer Berlin Heidelberg, 2005.

[3] H. Naz, M.N. Khokhar, *Critical Requirements Engineering Issues & their Solutions*, In Computer Modeling and Simulation, ICCMS'09, International Conference on, pp. 218-222, IEEE, 2009.

[4] A.M. Hickey and Davis, *Requirements elicitation and elicitation technique selection: a model for two knowledge-intensive software development processed*, In Proceedings of the 36th annual Hawaii international conference on system sciences, pp. 96-105, 6-9 Jan, 2003.

[5] B. Boehm, *Requirements that handle IKIWISI, COTS, and rapid change*, *Computer*, Vol *33*, Issue 7, pp. 99-102, 2000.

[6] L. Cao and B. Ramesh, *Agile requirements engineering practices and challenges: an empirical study*, Information Systems Journal, Vol 20, Issue 5, pp. 449-480, September 2010.

[7] L. Jun, W. Qiuzhen and G. Lin, *Application of Agile Requirement Engineering in Modest-sized Information Systems Development*, Second WRI World Congress on Software Engineering, pp. 207-210, 2010.

[8] M. Fowler, and J. Highsmith, *The agile manifesto,* Software Development Magazine 9.8 (2001): 28-35, 2001.

[9] B. Boehm and R. Turner, *Balancing agility and discipline: A guide for the perplexed*, Addison-Wesley Professional, 2003.

[10] M. Cohn and D. Ford, *Introducing an agile process to an organization*, Computer, *36*(6), pp. 74-78, 2003.

[11] P.E. McMahon, *Bridging agile and traditional development methods: a project management perspective,* CrossTalk: The Journal of Defense Software Engineering, May 2004.

[12] S. Nerur, R. Mahapatra, and G. Mangalaraj, *Challenges of migrating to agile methodologies*, Communications of the ACM, 48(5), pp. 72-78, 2005.

[13] Y. B. Leau, W.K. Loo, W.Y. Tham and S.F Tan, *Software Development Life Cycle Agile vs Traditional Approaches*, International Conference on Information and Network Technology, vol. 37, pp. 162-167, 2012.