Volume 4, Issue 12, December 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Lean Software Development – Survey on Agile and Lean Usage in Small and medium Software Firms in Bangalore

[1]Piyush Kumar Pareek, [2]Dr. A. N. Nandakumar
[1]Assistant Professor, Department of CSE, Kammavari Institute of Technology, Bengaluru & Research Scholar, Jain University, Bengaluru, India
[2]Principal, R.L.Jalappa institute of Technology & Research Guide, Jain University, Bengaluru, India

Abstract— Work done in software development is not value added work but is work caused by delays and interruptions. Working from old requirements, re-working requirements, the time to find bugs, thrashing due to integration, implementing misunderstood requirements often account for more than half of the work done by a development team. Delays in, and interruptions to, our workflow is mostly due to working on too many, too large projects. We must create visibility throughout the value stream so we can improve the efficiency of the work being done. We must manage the workflow across teams when more than one team is involved. The objective of this paper was to bring out more generalizable and up to date results on the state of agile software development and lean software development.

Keywords— Agile software development and Lean software development

I. INTRODUCTION

Origins of the term waste

The concept of "waste" in business was coined in the manufacturing industry during late 1940s. It was used by Toyota. In those days, automobiles involved a huge amount of manufacturing, and companies had to charge their customers a high price. The only option for Toyota to reduce car prices was to find ways to reduce the manufacturing costs. As part of this exercise, they started identifying "waste," which meant the feature (or process step) that did not add value for the customer. Once they identified the waste, they created ways to eliminate that waste from the system. Over time, "eliminate waste" became the fundamental concept of Lean, which was used in the manufacturing industry.

What is waste?

Waste is, in fact, the opposite of *value* (a capability delivered to the customer through which the customer attains a tangible or intangible benefit). So whatever feature or functionality or process step that neither adds value nor is used will be considered waste and should be eliminated from the system/product/process.

The seven wastes listed below were identified in manufacturing in the Toyota Production System (TPS) by Shigeo Shingo:

1. *Inventory:* Unfinished goods (also called as "work in progress," or WIP)
2. *Overproduction:* Producing more than the demand requires
3. *Extra processing:* Additional steps in the process that aren't really needed
4. *Transportation:* Shipping the goods from one place to the other
5. *Waiting:* Lag between process steps
6. *Motion:* Moving around within the process
7. *Defects:* Flaws in the deliverables that impact their features/functionality

List of wastes in SDLC phase wise:

WATERFALL MODEL:

Requirements Phase/Design Phase for developers Test Plan for Testers/Development Phase(Developers) and Test Suite development Phase(Testers)/Testing and Defect fixing phase/Release and Maintenance

ITERATIVE MODEL: It is same as Waterfall, besides entire SDLC is divided into small chunks where every chunk is treated as a waterfall model, and after every chunk is prepared it is integrated with the other chunks and the unsolved defects are moved on to the next phase.

AGILE MODEL: The deliveries can be as fast as weekly twice. The TIME TO MARKET is the least in this model, and this is the model which tries to create the minimum of wastes.

Now in iterative model, which was very widely used till 3 years back, we used to make a small POC(Proof of Concept), give a demo to the clients and then on top of that would start working on it. The reason I am having so much trouble in

providing you this information is probably due to the fact that my entire career had been spent in service sector where no softwares are developed from scratch. The deliveries and models are business oriented. These models contributes to lot of wastages

Most of the projects would spend time and resources to devise plans so that they would make a CASH COW of the clients. Thats why incase of any crisis, preventive measures are taken, but Corrective one are not taken. The frameworks are created to accommodate the changing technology and business model for now… but nothing is created to make it compatible with all future updations. Also another major reason for wastage are when a resource who is hired, trained and performing very nicely in a company and the same resource who has become a CRITICAL resource for a team, has finally decide to QUIT the organisation when it is most required. At that time, lot of time would parallely get wasted in finding and bringing the resource in sync with the current position of the project. Also availability of proper training resources is a must. If you look at the APIs for Java or SQL and look at it with the perspective of a NOVICE, you may find that those website, seem more like features which are difficult to understand, rather than easier to understand.

I've discussed this with people who work in construction. When they build an estimate they have known hour numbers for various small tasks. Every small task has been done millions of times before and has low variability. Every large task can be eventually broken down to some tally of these smaller, known tasks. Their estimates are very precise. The longer the overall project the more risk there is of the unknown interrupting this number, but this too is a straight-forward formula of padding.

In programming, if a developer can break down everything in a project into a set of tasks they've done many times before, they're doing a bad job. Everything in programming can be automated, so if you've done it enough before that you have that strong an understanding of it, you should be automating it by now. How long will it take to automate?

Reasons for Delays

1. It's difficult to meet the project deadlines when you are not consulted, or when the people making the decisions are not actively involved in software development.
2. Something became more complicated than it should have been. E.g. Tuio Touch Events not properly propagating to children of stage objects means that I now have to completely redesign the class based system that I had originally implemented for memory management reasons to be more flat.
3. Things were taking longer than expected, and I had to stop to make sure I get work for all my other classes completed
4. Sometimes a project might have some hidden problems that are genuinely difficult to solve, just from an algorithmic point-of-view. However, mostly, in my experience, projects fall behind because of factors external to the programming, such as non-robust platforms/frameworks/environments, problems in defining scope, requirements that are not clear and so on.

II. RESEARCH METHODS USED DURING SURVEY

Extensive Exploratory web survey study including almost 25 questions conducted among Small and medium Software Firms in Bangalore, 500 software engineers from almost 50 Small and medium Software Firms participated in this.

Data Analysis was done using SPSS Version 21.0

III. DATA ANALYSIS

Table 1 : My Role in the Company

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Developer	100	20.0	20.0	20.0
	Project Manager	30	6.0	6.0	26.0
	IT Staff	30	6.0	6.0	32.0
	Architect	30	6.0	6.0	38.0
	Consultant / Trainer	35	7.0	7.0	45.0
	Quality assurance / Tester	65	13.0	13.0	58.0
	Operations / Supporting Staff	35	7.0	7.0	65.0
	Scrum Master	35	7.0	7.0	72.0
	Process Manager	35	7.0	7.0	79.0
	Product Manager	35	7.0	7.0	86.0
	President / CEO	35	7.0	7.0	93.0
	Sales / Marketing Personnel	35	7.0	7.0	100.0
	Total	500	100.0	100.0	

In Table 1 : Software Developer who responded to questionnaire constitutes 20 % of the respondents and Quality Assurance / Testers Constituted 13 % of the respondents

Table 2 : Experience in Software Development

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	None	42	8.4	8.4	8.4
	Less Than 2	84	16.8	16.8	25.2
	2-5	126	25.2	25.2	50.4
	5-10	166	33.2	33.2	83.6
	10-20	41	8.2	8.2	91.8
	More than 20	41	8.2	8.2	100.0
	Total	500	100.0	100.0	

In Table 2 : The experience of the respondents between 2 – 10 Years constituted 58.4 % of the total participants.

Table 3: Size of the organisational Unit

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1-10	36	7.2	7.2	7.2
	11-50	36	7.2	7.2	14.4
	51-100	107	21.4	21.4	35.8
	101-200	107	21.4	21.4	57.2
	201-500	107	21.4	21.4	78.6
	501-1000	71	14.2	14.2	92.8
	More Than 1000	36	7.2	7.2	100.0
	Total	500	100.0	100.0	

In Table 3 : The size of the organizational unit of participants is represented,51 – 1000 Peoples organization constituted 77 % of respondents participation.

Table 4 : Usage of Agile and Lean Methods

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Only Agile	50	10.0	10.0	10.0
	Only Lean	200	40.0	40.0	50.0
	Agile & Lean	200	40.0	40.0	90.0
	No Agile or Lean	50	10.0	10.0	100.0
	Total	500	100.0	100.0	

In Table 4 : For the question on Usage of Agile and Lean methods in organization Usage of Lean was almost 80 % in these organization.

Table 5 : Usage of Specific Agile Practices

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Prioritised work list	49	9.8	9.8	9.8
	Iteration/Sprint Planning	24	4.8	4.8	14.6
	Daily Stand Up Meetings	24	4.8	4.8	19.4
	Unit Testing	24	4.8	4.8	24.2
	Release Testing	24	4.8	4.8	29.0
	Active Customer Participation	49	9.8	9.8	38.8
	Self Organising Teams	24	4.8	4.8	43.6
	Frequent and increment delivery of working software	24	4.8	4.8	48.4
	Automated Builds	26	5.2	5.2	53.6
	Continous Integration	26	5.2	5.2	58.8

	Contnous Integration	26	5.2	5.2	64.0
	TDD	26	5.2	5.2	69.2
	Retrospectives	26	5.2	5.2	74.4
	Burn-Down Charts	51	10.2	10.2	84.6
	Pair Programming	26	5.2	5.2	89.8
	Refactoring	26	5.2	5.2	95.0
	Collective code ownership	25	5.0	5.0	100.0
	Total	500	100.0	100.0	

From Table 5 we can make out Prioritised work list Active customer participation and Burn down charts are commonly used.

Table 6 :Usage of Specific Lean Principles

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Focus on creating customer value	75	15.0	15.0	15.0
	Eliminate waste and Excess activities	146	29.2	29.2	44.2
	Create a culture of continous improvement	27	5.4	5.4	49.6
	Do it right the first time	27	5.4	5.4	55.0
	Respect and Empower People	26	5.2	5.2	60.2
	Minimize inventory or work in progress	26	5.2	5.2	65.4
	Pull from Demand	26	5.2	5.2	70.6
	optimising the whole system	26	5.2	5.2	75.8
	continous flow of small batches	26	5.2	5.2	81.0
	Make decisions as late as possible	24	4.8	4.8	85.8
	Root source analysis is done after problems are discovered	24	4.8	4.8	90.6
	Look simultaneously for multiple solutions	23	4.6	4.6	95.2
	create trusted relationships with suppliers	24	4.8	4.8	100.0
	Total	500	100.0	100.0	

Table 6 represents usage of specific Lean Principles used in the organisation; It can be observed that Focusing on customer value and Elimination of waste is widely adopted.

IV. RESULTS

1. Respondents were mainly Developers and Quality Testers
2. Experienced on an average of 4.8 Years
3. Organisational Unit size was Medium Level Enterprises.
4. Strongly indicating usage of Agile and Lean Practices prevailing in small and Medium Level firms
5. Usage of specific Agile Practices like Prioritised work list Active customer participation and Burn down charts are commonly used.
6. Usage of specific Agile Principles : Focusing on customer value and Elimination of waste is widely adopted

V. CONCLUSION

1. The Software Market Is Becoming More Dynamic Which Can Be Seen In Frequently Changing Customer Needs
2. Software Companies Need To Be Able To Quickly Respond To These Changes.
3. This Means That They Have To Become Agile With The Objective Of Developing Features With Very Short Lead-Time And Of High Quality.
4. A Consequence Of This Challenge Is The Appearance Of Agile And Lean Software Development.

5. Lean Software Development Aims At Systematically Identifying Waste To Focus All Resources On Value Adding Activities.

REFERENCES

[1] B. Boehm and R. Turner, *Balancing Agility and Discipline. Boston,* MA: Addison-Wesley, 2004.
[2] T. Stober and U. Hansmann, *Agile Software Development: Best Practices for Large Software Development Projects. Berlin, Germany: Springer*- Verlag, 2010.
[3] M. Cohn, *Succeeding With Agile: Software Development Using Scrum.* Boston, MA: Addison-Wesley, 2010.
[4] K. Tate, *Sustainable Software Development: An Agile Perspective. New* Jersey: Addison-Wesley, 2006.
[5] J. Shore and S.Warden, *The Art of Agile Development. Sebastopol, CA:* O'Reilly, 2008.
[6] Bjornvig, G. & Coplien, J. (2010), *Lean Architecture: for Agile Software Development,* Hoboken: Wiley
[7] Poppendieck, M., & Poppendieck, T. (2009), *Leading Lean Software Development:* Crawfordsville: Addison-Wesley Professional

