Implementing Lean Thinking in Software Development – A Case Study from India

Gopalakrishnan Narayanamurthy
Fellow Program in Management
Quantitative Methods & Operations Management (QM & OM) Area
Indian Institute of Management Kozhikode (IIMK)
IIMK Campus, Kunnamangalam, Kozhikode, Kerala – 673570, India.
E-mail: gopaln06fpm@iimk.ac.in
Phone: +91-8943687765

Anand Gurumurthy *
Associate Professor
Quantitative Methods & Operations Management (QM & OM) Area
Indian Institute of Management Kozhikode (IIMK)
IIMK Campus, Kunnamangalam, Kozhikode, Kerala – 673570, India.
E-mail: anandg@iimk.ac.in
Phone: +91-495-2809435
Fax: +91-495-2803010

Shyam Prasath B
Post-graduate Program in Management
Indian Institute of Management Kozhikode (IIMK)
IIMK Campus, Kunnamangalam, Kozhikode, Kerala – 673570, India.
E-mail: shyamprasathrao@gmail.com
Phone: +91- 9741400893

* Corresponding Author

Biographical Statements

Gopalakrishnan Narayanamurthy is pursuing his Fellow (Doctoral) Program in Management in Quantitative Methods and Operations Management (QM & OM) in Indian Institute of Management Kozhikode (IIMK), Kerala, India. He is a recipient of Fulbright-Nehru Doctoral Research fellowship and worked at Carlson School of Management, University of Minnesota during his fellowship. His research focuses on assessment of process improvements in services.

Anand Gurumurthy is an Associate Professor in the area of "Quantitative Methods and Operations Management (QM&OM)" at the Indian Institute of Management, Kozhikode (IIMK), Kerala, India. Earlier, he was an Assistant Professor with the Mechanical Engineering Department of Birla Institute of Technology & Science (BITS) Pilani, Pilani Campus, Rajasthan, India, where he also completed his PhD in the area of Lean Manufacturing and ME in Manufacturing Systems Engineering. He received his BE in Mechanical Engineering from the University of Madras, India. He has around 12 years of teaching/research experience. He started his career as a Production Engineer with one of India's leading industrial houses – the TVS Group. He has published around 40 papers in peer-reviewed national and international journals such as International Journal of Production Research, and Production Planning & Control. He has also presented many papers in various national/international conferences. His current research interests include lean thinking, application of lean thinking in other sectors, operations management and world-class manufacturing.

Shyam Prasath B, currently works with Wipro Technologies as a Business Analyst in Bengaluru, India. Shyam has a PGDM from IIM Kozhikode and is an ardent researcher in the field of Software Development Process. He has a good blend of technology expertise and business acumen that help in studying the best practices followed in the industry in this domain and correlating them with the business needs of the industry and effectively contributing to the research in this area.

Implementing Lean Thinking in Software Development – A Case Study from India

Abstract: Implementation of lean thinking (LT) in the service sector has been widely reported. Although few studies describing the application of LT in software development (SD) are available, not many are from an emerging market such as India. Our study addresses this gap by using a single-case study methodology to understand the lean approach adapted by a firm in India to overcome the issues faced in its SD process. Data were collected through direct observation for a period of 1 year. Difficulty in integrating work from various teams, long release cycles for the developed software products, late shipments, quality issues, customers' dissatisfaction, and high operational costs were the problems faced by the case company. These problems motivated the case company to adopt LT at the team level by following the scrum process. This study identified how the LT approach guided the case company to achieve responsiveness, regular interaction between employees, involvement of customers, accomplishing targets within the planned timeline and so on. This study helps both academicians and practitioners to understand the approach followed to implement LT in a SD firm in India.

Keywords: Lean thinking; software development; lean services; case study; India.

Implementing Lean Thinking in Software Development – A Case Study from India

1. Introduction

According to the "IT-BPO Sector in India: Strategic Review 2012" report developed by India's National Association of Software and Services Companies, India's export of outsourced software product development (SPD) crossed the billion dollar mark in FY2011. The report also stated that the market for outsourced SPD had consistently experienced double-digit growth rates over the past 5 years and is expected to grow faster than the industry average of 17% to cross USD 1.2 billion in exports. Figure 1 captures the growth of information technology (IT) market in India at different time points.

"Insert figure 1 here"

A noticeable shift in the Indian software product business ecosystem is being observed, in terms of acceleration in software product business activity, improvements in the talent and support and innovations in software product technology and delivery/business models. Apart from these, changes in the Indian economy are also helping in the development of the domestic market for software products. Increasing IT adoption in India has helped in creating a sizeable product business market opportunity locally. For instance, the market worth for Indian domestic software products segment is pegged at Rs. 180 billion in FY2012, a growth of nearly 13% over FY2011 which can be attributed to the need to replace legacy systems, technology advancements around cloud, mobility and so on.

Some of the key advantages that promote the growth of software industry in India are the availability of abundant resources of science and technology talent, low cost of labor, high English proficiency among people, geographical advantages (12-hour time difference between India and the USA) and the supportive policies of the Government of India. Some of the disadvantages faced by the software industry in India are mentioned to be the lack of core technology and intellectual property rights, over-dependence on the world market (especially the USA) and serious insufficiency of IT industrial infrastructure. The above-mentioned facts about India's SPD market clarify the importance of carrying out this study from an Indian context.

Post setting up the SD organizations in India to utilize the key advantages of the market, these firms face a variety of problems due to the inherently dynamic nature of the software market. SD business has been affected by frequently changing customer needs and rapid evolution of technology, which place these firms under immense pressure to deliver their outputs within the prescribed time frame at the agreed upon quality and cost. To remain competitive, companies are continuously attempting to react to the changing needs in an agile manner (Poppendieck and Poppendieck 2010). Not meeting up with the rapidly varying demands of the market would result in a higher risk of market lockout and reduced probability of market dominance. Hence, many of the Indian software organizations are attempting to maintain their competitive advantage in the global platform by reorganizing their SD processes (SDPs) by introducing improvements in the conventional SD life cycle models (such as waterfall model, incremental model and spiral model). Recently, Indian software organizations have been exploring the possibility of introducing the principles and concepts of lean thinking (LT) proposed by Womack et al. (2003) to improve the SDP. LT has been observed to have the capability to yield a significant reduction in cost and variability while improving the quality level and flexibility. In this study, an attempt has been made to understand the procedure of implementing lean SD (LSD) followed by an Indian case company. Figure 2 shows the steps adopted in conducting and reporting this study.

"Insert figure 2 here"

2. Literature Review

In this section, literature review of LSD is performed in two directions: (a) recent literature of LSD and (b) case studies on LSD. The review of recent literature of LSD was carried out to understand the extent of implementation of LT in the domain of SD and also to identify the prerequisites, practices, principles and performance measures adopted. Case studies on LSD was reviewed separately as the current study is contributing to this literature.

2.1. Recent literature of LSD

In this section, the reviewed literature has been classified into the following categories (Anand et al., 2014): value and waste, lean practices in SDP and specialized tools and techniques (as categorized in Table 1). This review process clearly showed that the studies lacked a detailed description on the LT adoption procedure and none of the studies have been performed in the Indian context. Probable reasons could be that most of the Indian software organizations act as "outsourcing partners" or "service providers" to other big organizations in the USA. Due to data

security and confidentiality clauses in the agreements, many organizations are forced not to share their implementation journey.

"Insert table 1 here"

2.2. Case studies on LSD

In this section, the literature on case studies in LSD were reviewed to understand the aspects of LSD from the perspective of a firm. Table 2 suggests that the implementation of LSD has started gaining importance among the practitioners, although the concept of LSD is still evolving.

"Insert table 2 here"

2.3. Research gaps

Reviews documented in Sections 2.2 and 2.3 indicate that a good number of research papers and books dealing with the theoretical aspects of applying LT to SD are available in the literature. Researchers have attempted to identify the wastes that happen in SD and suggested different tools to identify and reduce/eliminate wastes. Other researchers have attempted to document the implementation aspect of LT in SD through case studies. However, almost all the studies are based on firms in the Western context (except that of Staats et al., 2011), even though India is a preferred market for outsourcing SD. The study of Staats et al. (2011) clearly indicates that implementation of LSD is happening in Indian software organizations. However, the objective of their study was to understand the impact of LT on performance improvement within the chosen case organization and was not to report how the case organization was implementing LSD, which is the objective of the current study.

3. About the Indian case organization

The chosen case company is a leading provider of global enterprise software, that is, business software solutions, and it has a major SD facility located in India. The global company has been in existence for more than 40 years and its SD facility in India has been in existence for more than 17 years. The current employee strength is approximately 82,000 globally, and the number of employees in its Indian development centre is approximately 6000. It has offices across 130 countries in the world, which are split into four geographical regions: EMEA (Europe, Middle East, Africa), APJ (Asia Pacific, Japan), America (the USA and Canada) and LAC (Latin America and Caribbean). The company caters to a number of industry verticals such as aerospace

and defence, automotive, banking, consumer products, oil and gas, engineering and construction, health care, media, public sector, retail and telecom. The company implemented LT across the entire SDP and also for its support teams, and it clocked its highest revenue since inception after 3 years of LT roll-out. This could be partially attributed to the new products that were introduced as well as to the LT that was implemented in the company. In this study, a detailed description of the procedure adopted by the case company to imbibe the LSD approach is discussed. Data were collected through direct observations and experiences over a period of 1 year by one of the authors.

3.1. Drivers for Implementing LT

The implementation of LT at the case organization was initiated after it was found that it was becoming increasingly cumbersome to integrate the work from various teams. The release cycles of the products were long and were split into many phases. Issues in output quality, frequent shipment delays, missed/unnoticed needs of the customers, and high operational cost were the problems faced. The average release cycle of a software product was 6 months, and within this period, there were drastic changes in the requirements of industry and customer. For example, a simple search feature was ordered by a customer at the initial stages of development, but at the end of 6 months while delivering the product, due to changes in the requirements and variations in the market dynamics, the customer incrementally modified the requirement to a full-text search feature. The case organization had to improve its responsiveness to accommodate such dynamic changes. The company was facing difficulties in satisfying its customer's bug ratio specifications mentioned in the service-level agreement. For a fundamental error committed at the initial stages of SD, the number of valid bugs increased exponentially by the end of 6 months as the error had a cascading effect in generating the bugs with progressing in SD.

To overcome these problems and attain a competency, the case organization started the roll-out of LT in 2006. The adoption took place when the company was changing its business strategy. The company had identified new domains to invest for the future and was coming up with new software products for each of the new domains. Company adopted LT across the entire software product value chain and insisted on following the methodology for all the future new software products. Teams working on the new software products were trained on a fast-track basis. Fast-track training helped in setting the benchmark for other teams and also in conveying the seriousness of the LT initiative undertaken by the company.

To provide in-depth details and clarity about the LSD approach adopted in the case organization, both the organizational-level and the project team-level changes and adaptations are reported in this study. A project team working on developing the "search" software in the case organization is chosen for reporting the process adopted. The team comprises of 10 members catering to three functionalities: quality engineering and testing (QET), performance, and production. QET and performance were the main functionalities, and production was a support function. QET function checked the entire product functioning through feature testing, whereas performance function would carry out load testing with heavy data for measuring the response time. Production function catered to QET and performance functions by performing installations, data loading, and other support functions. Current problems faced, associated wastes and LSD practices as solutions to the problems identified are listed in Table 3.

"Insert table 3 here"

4. LSD Implementation

4.1. Communication

The entire company was informed about the importance of adopting LT through e-mails, town halls and other business unit-level meetings. The training was made mandatory for all the employees. Constant meetings and demo sessions were conducted to explain how LT can be adapted in each and every team. Specific days were reserved to conduct tutorial sessions on lean principles, and subject-matter experts were invited to attend the sessions. On average, 60 employees spread across multiple teams and multiple products were trained in a period of 1 month. Different teams were formed with each team essentially comprising a solution owner, a product owner, a line manager and a dedicated scrum master. Individual roles of team members are detailed in Section 4.2. Each "search" software project team member underwent training for 2 weeks on lean and scrum.

4.2. Team Formation

Team comprises of three important players described below:

a) Solution owner: A problem or task from customers along with the inputs is given to the solution owner through various channels. Solving a problem might require one or more products existing in the portfolio. The solution owner identifies those products and then transfers the problem on hand to the respective product owners along with the completion deadline. The

solution owner is responsible for the communication with the external stakeholders regarding the solution in hand and coordinates with the product owners of various products.

b) Product owner: Every product in the company has a product owner. The level of granularity attributed to the product is high, and even a small application in a large software may be classified as a product sometimes depending on the quantum of the work involved in its development and maintenance. For instance, a search feature in an enterprise resource planning is considered as a product; it has a separate product team and its product owner is responsible for all the activities regarding the product. Usually, such teams are formed at the inception of the product and expertise is developed on the product from then on. These teams expand slowly and steadily based on the growth of features in the product. The role of a product owner is highly diverse ranging from project management to technical program management. The product owner heads the cross-functional team that specializes in all parts of SD such as development, testing, production and performance. He/she is accountable for the quality and shipment of the product.

c) Line manager: Line manager is typically appointed to monitor two to three teams, and he/she is a homegrown member with good amount of experience (5–8 years) in monitoring the team performance. The role of a line manager in a scrum team is that of a facilitator. For example, if a team member needs to coordinate with external or internal stakeholders from other teams for knowledge transfer, it is facilitated by the line manager. Line manager is held responsible for arranging the required resources including hardware, arranging for training, authorizing leaves and other human resource activities such as appraisals, performance ratings and bonus payments. The line manager plays very minimum role in the actual scrum process, but every update in the team goes through him/her. Line manager is aware of every deliverable and its progress with the current status. He/she also handles the finance of the team and monitors all the costs incurred. Line manager reports to the program director of the business unit, who in turn reports to the vice president of the business unit. “Search” software project team comprises of 10 team members with a product owner, a solution owner, and a line manager, who have multiple projects functioning under their purview.

4.3. Micro Level Lean Implementation

Implementation of LT is attempted at the team level through a process called scrum (as shown in Figure 3). Scrum is a methodology that dictates the way in which every team member has to work and gives the broad guidelines for deliverables along with the timelines (Poppendieck &

Poppendieck, 2003; Vlaanderen et al. 2011). Figure 4 shows in detail the scrum procedure starting from the release plan to the final shipping of the product.

"Insert figure 3 & 4 here"

"Search" software team consisting of 10 members is divided into three scrum teams catering to three individual functionalities mentioned earlier (as shown in Figure 5). Scrum teams are formed such that the common team members prevail across the functionalities. Splitting the three functionalities into three scrum teams reduces the total time spent by each individual in meetings as initially all the team members have to sit for every meeting irrespective of their task relevance to that meeting. In this scenario, meetings are specific to functionality teams, and only members (both specific and common) belonging to those functionalities attend the meeting, thereby providing more aggregate time to each individual to work on their task. The common members of the functionalities share the relevant knowledge and updates from their scrum meeting to reduce the information asymmetry between the functionalities.

"Insert figure 5 here"

Scrum team: Every scrum team has a scrum master, who is ideally the most senior person in the team, to coordinate the scrum process (Rising & Janoff, 2000). The scrum master has a thorough knowledge of the list of deliverables that is updated every month and the approximate time available to complete the deliverables (Marchesi et al., 2007). The backlogs (smallest unit of the job) are released every month by the scrum master for the entire team along with the expected completion timeline. The team members choose the backlogs based on what they can finish in a sprint, which is mostly a month long. Depending on the team's composition and the nature of work, the team can also choose to have a 2-week sprint. All the three scrum teams with their scrum masters decide to follow the monthly sprint as it would help in synchronizing the project flow. This decision is updated to the top management with complete details, including the meeting frequency, meeting day in a week and timing in a day.

Scrum process: The scrum process consists of multiple sprints/takts (Rising & Janoff, 2000; Hossain et al., 2009). The requirements that are communicated to the product owner from the solution owner are broken down into pieces as final deliverables and communicated to each team. Internal project monitoring tools track the uploaded backlog list of the product owner, individual

backlog progress and final sprint completions. Every backlog is assigned a status based on its position in the scrum process. Five statuses given in general are unassigned, assigned, in progress, blocker and completed. The blocker status is assigned to a set of backlogs that cannot be completed because of non-availability of required software/hardware resources or skills. Once a team member is allotted to perform a task, it is his or her responsibility to assign a suitable status to the tasks from then on. Every scrum team maintains a separate interface with tools similar to the project monitoring tool and is accessed by the team members. Administrator rights are provided with the scrum master and product owner who can assign suitable access rights to the team members. In the "search" software project being discussed, the progress of the three individual scrum teams and individual team members are displayed to the entire "search" software project team to coordinate and operate in synchronization.

Process flow: In the beginning of every sprint, the product owner releases the list of deliverables to each scrum team. A sprint planning meeting is held and all the teams under a product owner, that is, the cross-functional team under that product owner, are part of it. Tasks are assigned to the team members based on their availability and the requirements of the tasks. In the sprint planning meeting, the dependencies that each member have on each other are discussed and who has to assist and coordinate with each other is also identified. At the end of the sprint planning meeting, a list of deliverables/backlogs taken up for that sprint is finalized and the ones that could not be taken up are left for the next sprint. After finalizing the list of deliverables, tasks and activities to achieve the target are taken up at the scrum team level. For example, a list of deliverables that need to be submitted in one particular sprint by the QET scrum team of "search" software project include data type validation, scenario analysis of full-text search and automated script generation for model creation. These deliverables are assigned by the scrum master to the QET scrum team by choosing the backlogs provided by the product owner after attaining the consent from the scrum team members.

Daily scrum meeting: Ten-minute meeting or the daily stand-up called the scrum meeting is conducted every day (Fitsilis, 2008). It is compulsory for every team member to be present at the meeting. The team members stand in a semicircle facing each other with a scrum board in the middle, which typically is a whiteboard filled with sticky notes detailing the backlog names and their statuses (as shown in Figure 6). Over time, backlogs are grouped based on their status. In the daily scrum meeting, every team member discusses the backlog he/she has been working on the previous day, its current status and the working agenda for the current day, and finally

indicates the expected date of completion of the remaining tasks in the backlog. This meeting gives an opportunity for every member to voice his/her issues being faced in completing the backlogs. For instance, when a team member experiences any blocker, it should be brought to the attention of other team members. This kind of short meetings helps in finding a person with the experience to solve the problem. For example, any other team member with past experience and know-how on similar issues suggests potential solutions or approaches to solve it, thereby making the prior knowledge available to the team members easily rather than spending time in researching on the problems that have been already solved.

"Insert figure 6 here"

The number and time zone of global teams involved from different locations for solving the backlogs affect the quantum of work and the frequency of scrum meeting. This is a customization that is left to the prerogative of the scrum team. The multi-location teams typically have two scrum meetings a week via teleconferencing. The product owner needs to be part of the everyday scrum meeting to monitor the progress of the deliverables and give inputs from time to time depending on the updates from the external stakeholders.

Sprint review: On the last day of every sprint, a sprint review meeting is held and this time all scrum teams report to the product owner (Schwaber, 1997). In this meeting, each and every deliverable committed for the sprint is taken up and discussed. The update is given by each member on his/her deliverables, and if the backlog remains incomplete, the reason for the same is stated. For instance, a QET scrum team of the "search" software project had the backlog of incomplete full-text search. The backlog had a defect with respect to long text data type, and the issue was raised to the development team. The status of the issue raised by the development team was presented at the sprint review, and this moved the accountability for the backlog to the development team from the QET scrum team. Individual as well as team performances were explicitly visible to the product owner from the status of the backlogs assigned at the start of the sprint. At the end of this meeting, the product owner took up the completed tasks and committed them to the solution owner, and the incomplete tasks were brought back to the next sprint. After every sprint, there is a gap of 1–3 days to look back at the sprint and analyse what went wrong and how it can be further improved in the future. Several project-related suggestions and contacts of experts are mentioned in the sprint reviews to improve the lead time by completing the backlogs with zero delays.

Sprint retrospective: In this meeting, every team member shares his/her tacit knowledge learned and discusses what went right or wrong in his/her approach towards a deliverable (Marchesi et al., 2007). Any other member with the knowledge of rectifying the problem communicates it to the entire team. This provides the opportunity for a new member to speak up about his/her difficulties and seek knowledge transfer from another experienced member in the team on how to overcome the difficulties. The line managers are not part of this meeting so as to provide space for the members of the team to share their true opinions and suggestions without the worry of career growth in the organization. After completing the sprint retrospective, the product owner and the scrum master discuss to finalize the deliverables that need to be taken up for the next sprint, how they can be performed with the available resources, and the steps that need to be taken to incorporate the proposed suggestions. Deliverables chosen after the discussion are communicated to the solution owner. Necessary steps are also taken to organize any required training to impart skills essential for the deliverables chosen.

Continuous improvement process: Continuous improvement meetings are organized by the line managers individually for every scrum team at regular intervals, mostly once in two sprints (Sutherland et al., 2008). In these meetings, all the team members speak up about the whole scrum process and give their suggestions for improvement. For example, a suggestion was made by the team members that the sprint retrospective meeting was not turning out to be very useful. This suggestion was taken up, and from then on, its frequency was changed from once in a sprint to once in two sprints. Such suggestions made by the team members are taken up by considering the phase of the product life cycle. If the product is new and is undergoing a lot of changes, there is a lot of scope for information exchange and transfer of tacit knowledge; hence, the frequency of meetings for retrospective is kept high. Continuous improvement is seen as a double-edged sword by few of the team members. They feel that most of the time and efforts are spent on the meetings for continuous improvement, thus leaving very less time for implementing the improvements. Hence, a balance has to be drawn between the time allotted for meeting in the scrum process and the time allotted for performing the task.

Customizations and adaptations in lean: Every team does customization to the roles according to its workload and time constraints. For instance, according to the actually mandated process, the scrum master should not be part of the task of deliverables. However, in reality, the scrum master is typically the team lead who has been in the team since its inception and knows the

product along with the quantum of work involved in every deliverable. His/her role is analogous to that of a supervisor or operations manager on the shop floor of a manufacturing company adopting lean principles. Therefore, this puts an additional workload on the team leader to coordinate the entire scrum process along with the regular tasks performed earlier. This also gives the team members a scope to stay abreast with the new challenges faced by the team and also learn in the due process.

The shipment is always done with an expectation to improve and resend the product perfectly in the next shipment of the subsequent sprint. As described, the product manager ships deliverables irrespective of its full completion (provided the bugs are within the limits prescribed), thereby providing a chance for the customer to review the shipped product and provide immediate feedback which can be incorporated by the developer in the subsequent sprints. This ensures frequent interaction between the customer and the developer, and helps in correcting the defects as and when they occur without carrying them forward to the finished product, which in turn would demand larger rework for rectification. This practice also helps in avoiding misunderstanding and deviations from the specifications.

5. Results and Discussion

The experience of implementing LT had a complete transformation as it imparted a sense of agility in the minds of the employees. Transformation helped in bringing down the release cycle from 6 months to 4 weeks, and once in every 4 weeks, a new version of the product was shipped. This new system created a reformed thinking process for the employees to work under strict deadlines. Employees mentioned that they were working constantly towards the objective and felt more motivated and focused. Daily scrum meetings were analogous to the Kanban practice in the manufacturing sector, where every scrum employee updates the backlog status being worked on and the next backlog that he/she will be taking up. It was also noticed that LSD mandates customers to adopt the LT approach. For instance, the current case organization that ships the products at an interval of 2–4 weeks (sprint duration) would require the customers' team to respond within that specific time duration, which in turn would require them to adopt the LT approach.

LT practices applied in SD showed how the implemented solutions guided the case company to achieve improved knowledge transfer, regular interaction of employees, involvement of customers and thereby facilitate shipping under short deadlines. Table 4 captures the changes in

performance measures after implementation of LT. An increase in the bug ratio from 40% to 80% can be attributed to a significant decrease in the total number of issues raised (from 165 to 15) and a comparatively little decrease in the actual number of valid defects (from 66 to 12) after LSD implementation. On the contrary to the reduction of the redundant tasks, it was noticed that number of meetings, e-mails transacted, and tickets raised per team increased significantly. However, these redundant tasks helped in reducing the problems that were noticed in the beginning.

Though the company was found to improve in its performance measures as a result of adopting the LSD approach, there were few questions from the employees that lacked a clear answer. The questions were as follows: what should be the ideal set of actions when the planned deliverables are not met completely within the assigned time? Should the shipment be delayed until the deliverables are completed or should it be shipped in the present form with an agenda for fixing the leftover issues in the subsequent sprint? LSD is still evolving in the case organisation and is considered to be a potential tool to reduce non-value-adding activities and improve value-adding attributes to the end customer. Through the study of the case organization, it was clearly observed that scrum was attempting to implement pull production where customers were involved regularly at the end of the sprint, and their feedback along with subsequent requirements was accommodated in the subsequent sprint.

6. Conclusion

In the beginning of this paper, it was claimed that this study would address the gap of absence of studies describing the application of LT in SD from an emerging market such as India. Using the single-case study methodology, the LT approach that was followed by a software firm in India was enumerated. To answer the research question of how LT in Indian SD firms is being implemented, what impact it has on the SDPs and how it can be improved, empirical data were collected through direct observations. By understanding the data collected, the procedure followed by the case company to implement LT at the team level through a process called scrum was documented in detail. Insights were also drawn from the case study's experience on how LT approach can guide SD firms in achieving responsiveness, information transfer between employees, involvement of customers and targeted deadlines. This study provides a detailed report for researchers and practitioners to understand how an Indian company is adopting LSD. As this study is adopting the single-case study methodology, it lacks the generalizability of the inferences, and future study can overcome this limitation by empirically studying several other

organizations adopting LSD. Future studies can also attempt to generalize the insights and the impact of implementation of LSD in Indian firms by adopting a survey methodology.

References

1. Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on* (pp. 9-16). IEEE.
2. Alegría, J.A.H., Bastarrica, M.C. & Bergel, A. (2011). Analyzing software process models with AVISPA. *Proceedings of the ICSSP'2011*. Available at: http://www.bergel.eu/download/papers/Berg11a-icssp11.pdf (accessed on 20 July 2011).
3. Anand, G., Chandrashekar, A., & Narayanamurthy, G. (2014). Business process reengineering through lean thinking: a case study. *Journal of Enterprise Transformation*, *4*(2), 123-150.
4. Antinyan, V., Staron, M., Meding, W., Osterstrom, P., Wikstrom, E., Wranker, J., ... & Hansson, J. (2014, February). Identifying risky areas of software code in Agile/Lean software development: An industrial experience report. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on* (pp. 154-163). IEEE.
5. Bastarrica, M.C., Alegría, J.A.H. & Bergel, A. (2011). Toward lean development in formally specified software processes. *Proceedings of the 18th EuroSPI'11*. Available at: http://www.bergel.eu/download/papers/Berg11eLeanProcess.pdf (accessed on 20 July 2011).
6. Bocock, L. and Martin, A. (2011). There's something about lean: A case study. *Proceedings of the 2011 Agile Conference*, 8–12 August, Salt Lake City, Utah, pp.10-19.
7. Bosch, J., Olsson, H. H., Björk, J., & Ljungblad, J. (2013). The early stage software startup development model: a framework for operationalizing lean principles in software startups. In *Lean Enterprise Software and Systems* (pp. 1-15). Springer Berlin Heidelberg.
8. Cawley, O., Richardson, I. & Wang, X. (2011). Medical device software development - A perspective from a lean manufacturing plant. In O'Connor, R.V., Rout, T., McCaffery, F. & Dorling, A. (eds), *Software process improvement and capability determination* (84-96). Berlin: Springer.
9. Corona, E. & Pani, FE. (2012). An investigation of approaches to set up a kanban board, and of tools to manage it. In Niola, V., Kadoch, M. & Zemliak, A. (eds.) *Recent Researches in Communications, Signals and Information Technology*, pp. 53-58. Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS).
10. Curtis, B. (2011). Cutting IT costs by applying lean principles. White paper. Available at: http://www.castsoftware.com/castresources/materials/wp/leanapplicationmanagement.pdf (accessed on 20 July 2011).
11. Fitsilis, P. (2008). Comparing PMBOK and Agile Project Management software development processes. In *Advances in Computer and Information Sciences and Engineering* (pp. 378-383). Springer Netherlands.
12. Gautam, N. and Singh, N. (2008), "Lean product development: Maximizing the customer perceived value through design change (redesign)", *International Journal of Production Economics*, Vol. 114 No. 1, pp.313-332.
13. Gong, Y., & Janssen, M. (2014). The Use of Lean Principles in IT Service Innovation: Insights from an Explorative Case Study. In *Digital Services and Information Intelligence* (pp. 58-69). Springer Berlin Heidelberg.
14. Hossain, E., Babar, M. A., & Paik, H. Y. (2009, July). Using scrum in global software development: a systematic literature review. In *2009 Fourth IEEE International Conference on Global Software Engineering* (pp. 175-184). Ieee.
15. Iberle, K. (2010). Lean system integration at Hewlett-Packard. Proceedings of the *28th Annual Pacific Northwest Software Quality Conference*, 18–19 October, Portland, Oregon, USA, pp.187-204.
16. Ikonen, M., Pirinen, E. Fagerholm, F., et al. (2011). On the impact of kanban on software project work: An empirical case study investigation. *Proceedings of the 16th IEEE ICECCS* doi:10.1109/ICECCS.2011.37.
17. Janes, A., & Succi, G. (2014). Enabling Lean Software Development. In *Lean Software Development in Action* (pp. 129-148). Springer Berlin Heidelberg.

18. Kirovska, N., & Koceski, S. (2015). Usage of Kanban methodology at software development teams. *Journal of Applied Economics and Business*, *3*(3), 25-34.
19. Kundu, G., & Manohar, B. M. (2012). Critical success factors for implementing lean practices in it support services. *International Journal for Quality Research*, *6*(4), pp. 301-312.
20. Kundu, G.K., Manohar, B.M. & Bairi, J. (2011). IT support service: identification and categorisation of waste, *International Journal of Value Chain Management, 5*(1), pp. 68-91.
21. Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and Software Technology*, *62*, 143-163.
22. Kuusela, R. & Koivuluoma, M. (2011). Lean transformation framework for software intensive companies: responding to challenges created by the cloud. Proceedings of the *37th EUROMICRO Conference on Software Engineering and Advanced Applications* (SEAA 2011), 30 August - 2 September, Oulu, Finland, pp.378-382.
23. Malladi, S., Dominic, P.D.D. & Kamil, A. (2011). Lean principles in IT services: a case study on implementation and best practices, *International Journal of Business Information Systems, 8*(3), 247-268.
24. Mandic, V., Oivo, M., Rodriguez, P., Kuvaja, P., Kaikkonen, H. & Turhan, B. (2010). What is flowing in lean software development? In Abrahamsson, P. & Oza, N. (eds), Proceedings of the *1st International Conference on Lean Enterprise Software and Systems* (LESS 2010), 17-20 October, Helsinki, Finland, pp.72-84.
25. Mäntylä, M. V., Adams, B., Khomh, F., Engström, E., & Petersen, K. (2015). On rapid releases and software testing: a case study and a semi-systematic literature review. *Empirical Software Engineering*, *20*(5), 1384-1425.
26. Marchesi, M., Mannaro, K., Uras, S., & Locci, M. (2007, June). Distributed Scrum in research project management. In *International Conference on Extreme Programming and Agile Processes in Software Engineering* (pp. 240-244). Springer Berlin Heidelberg.
27. Middleton, P. & Joyce, D. (2011). Lean software management: BBC Worldwide case study. *IEEE Transactions on Engineering Management*, doi: 10.1109/TEM.2010.2081675.
28. Middleton, P. (2001). Lean software development: two case studies. *Software Quality Journal*, *9*, 241–252.
29. Middleton, P. Flaxel, A. & Cookson, A. (2005). Lean software management case study: Timberline Inc. In Baumeister, H., Marchesi, M. & Holcombe, M. (eds), *Extreme programming and agile processes in software engineering*, 1st edn. Springer, Berlin: Springer.
30. Mohan, K.K., Harun, R.S., Srividya, A. & Verma, A.K. (2010). Quality framework for reliability improvement in SAP netweaver business intelligence environment through lean software development–a practical perspective. *International Journal of Systems Assurance Engineering and Management, 1*(4), pp. 316-323.
31. Nord, R. L., Ozkaya, I., & Sangwan, R. S. (2012). Making architecture visible to improve flow management in lean software development. *Software, IEEE*, *29*(5), 33-39.
32. Norrmalm, T. (2011). Achieving lean software development: implementation of agile and lean practices in a manufacturing-oriented organization, Thesis, available via: http://www.utn.uu.se/sts/cms/filarea/1102_Thomas%20Norrmalm.pdf.
33. Pernstal, J., Feldt, R., & Gorschek, T. (2013). The lean gap: A review of lean approaches to large-scale software systems development. *Journal of Systems and Software*, *86*(11), 2797-2821.
34. Petersen, K. & Wohlin, C. (2011). Measuring the flow in lean software development. *Journal of Software: Practice and Experience*, *41*(9), pp. 975–996.
35. Petersen, K. & Wohlin, C. (2010). Software process improvement through the lean measurement (SPI-LEAM) method. *Journal of Systems and Software, 83*(7), pp. 1275-1287.
36. Petersen, K. (2012). A palette of lean indicators to detect waste in software maintenance: A case study. In Aalst, W. Mylopoulos, J. Rosemann, M. Shaw, M.J. & Szyperski, C. (eds), *Agile processes in software engineering and extreme programming* (108-122). Berlin: Springer.
37. Poppendieck, M. & Poppendieck, T. (2010). *Leading Lean Software Development: Results are not the point*, New Jersey: Addison-Wesley.
38. Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley.
39. Poppendieck, M., & Poppendieck, T. (2006). *Implementing Lean Software Development: From Concept to Cash*, USA: Addison-Wesley Professional.
40. Raman, S. (1998). Lean software development: Is it feasible? Proceedings of the *17th IEEE Digit Avionics System Conference*, doi: 10.1109/DASC.1998.741480.

41. Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE software*, *17*(4), 26.
42. Rodriguez, P., Mikkonen, K., Kuvaja, P., Oivo, M., & Garbajosa, J. (2013). Building lean thinking in a telecom software development organization: strengths and challenges. In Proceedings of the *2013 International Conference on Software and System Process,* San Francisco, pp. 98-107.
43. Rodriguez, P., Partanen, J., Kuvaja, P., & Oivo, M. (2014, January). Combining lean thinking and agile methods for software development: A case study of a finnish provider of wireless embedded systems detailed. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 4770-4779). IEEE.
44. Rudolf, H. & Paulisch, F. (2010). Experience Report: Product Creation through Lean Approaches. In Abrahamsson, P. and Oza, N. (eds), *Proceedings of the 1st International Conference on Lean Enterprise Software and Systems* (LESS 2010), 17-20 October, Helsinki, Finland, pp.104-110.
45. Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117-134). Springer London.
46. Staats, B.R., Brunner, D.J. & Upton, D.M. (2011). Lean principles, learning, and knowledge work: Evidence from a software services provider. *Journal of Operations Management, 29*(5), pp. 376-390.
47. Staron, M., Meding, W., & Palm, K. (2012). Release readiness indicator for mature agile and lean software development projects. In *Agile Processes in Software Engineering and Extreme Programming* (pp. 93-107). Springer Berlin Heidelberg.
48. Sutherland, J., Jakobsen, C. R., & Johnson, K. (2008, January). Scrum and CMMI level 5: The magic potion for code warriors. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual* (pp. 466-466). IEEE.
49. Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and software technology*, *53*(1), 58-70.
50. Wang, X., Conboy, K., & Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, *85*(6), 1287-1299.
51. Widman, J. Hua, S.Y. & Ross, S.C. (2010). Applying lean principles in software development process – a case study. *Issues in Information Systems*, *9*(1), 635-639.
52. Womack, J.P. & Jones, D.T. (2003). *Lean Thinking: Banish Waste and Create Wealth in YourCcorporation*, New York: Simon & Schuster.

Figures

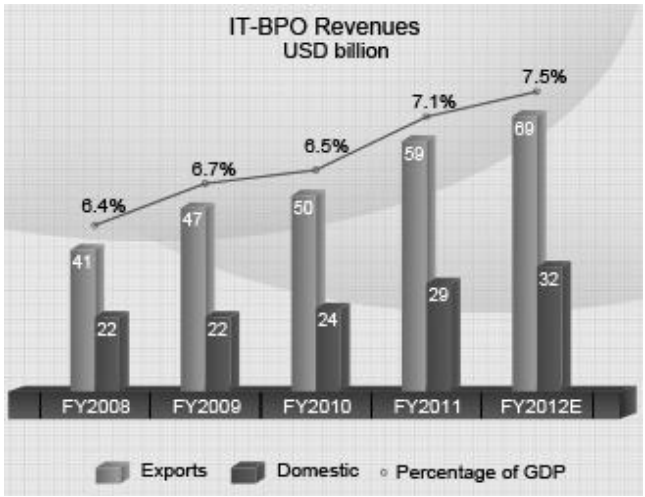

Figure 1: Composition of IT market in India: 2008-2012

(Source: In Nasscom website on Indian IT-BPO Industry)

Figure 2: Steps adopted in conducting the current study

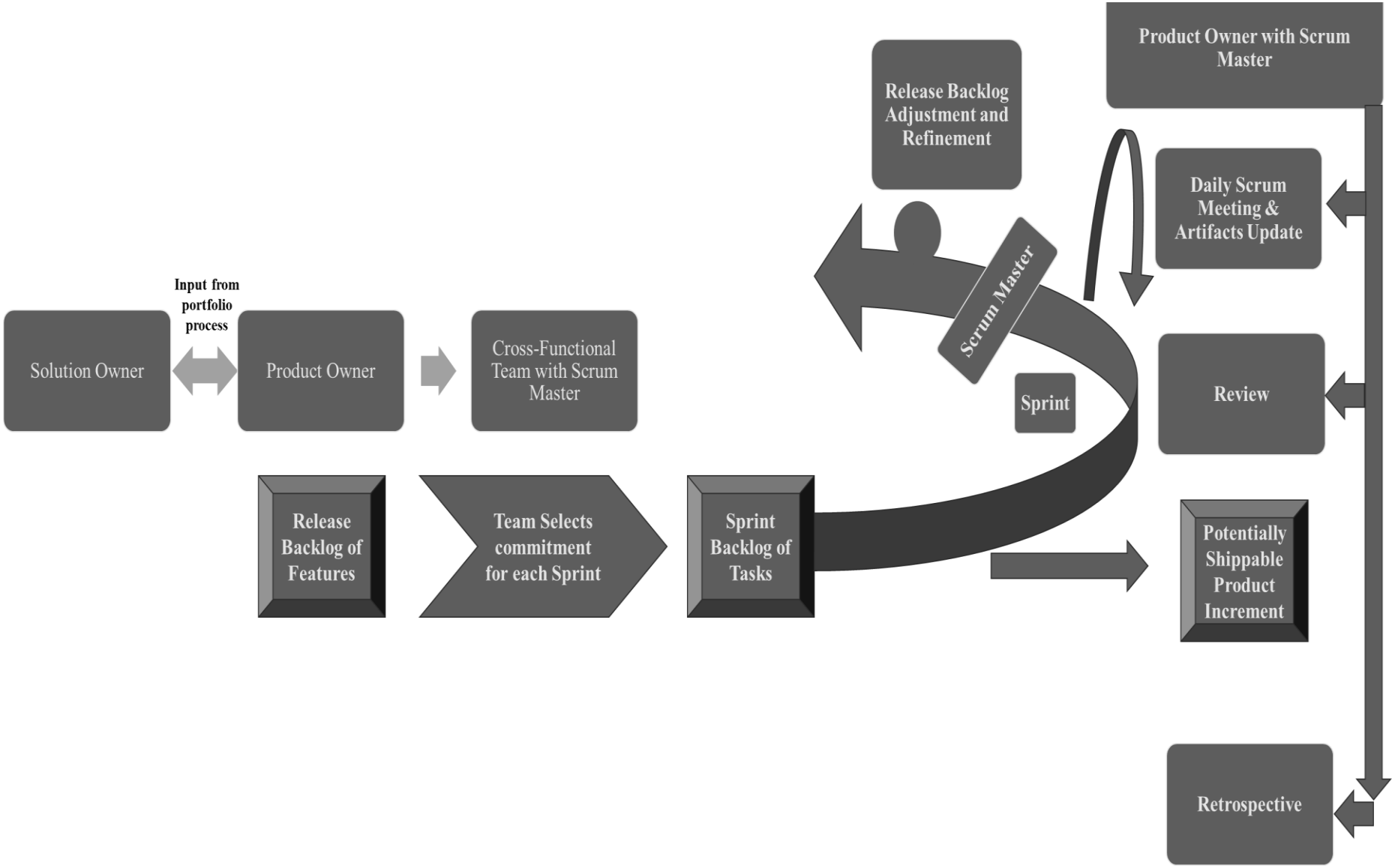

Figure 3: Flow in Scrum Process

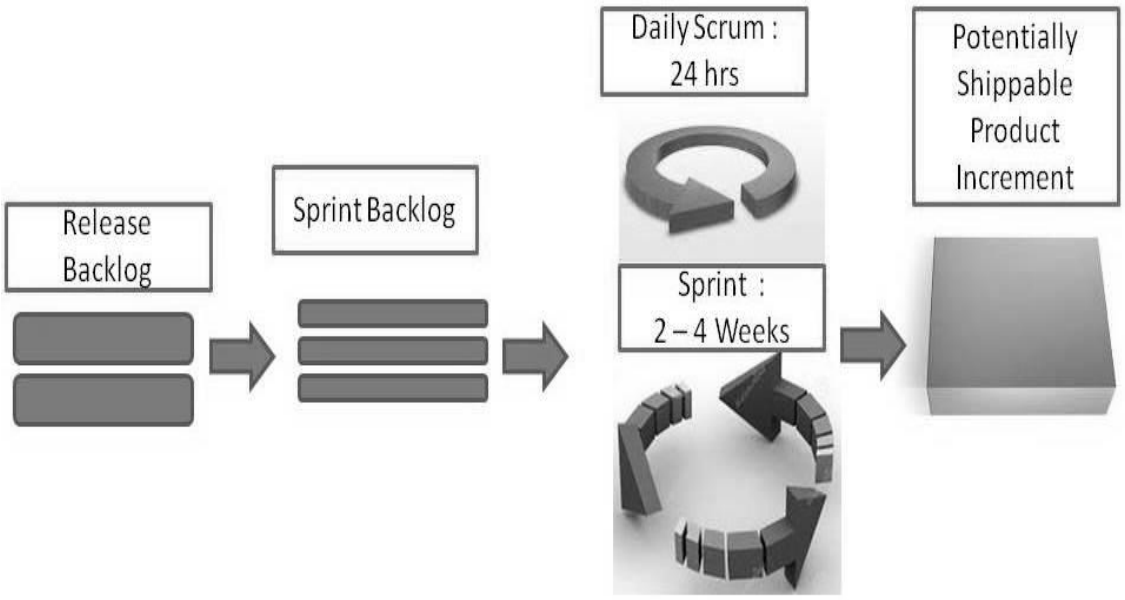

Figure 4: Snapshot of Scrum Process

Function	Specific	Common	Total
QET	4	QET & Performance - 2 QET & Production – 1	7
Performance	1	QET & Performance – 2	3
Production	2	QET & Production - 1	3

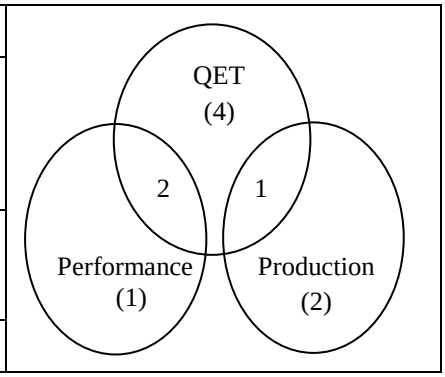

Figure 5: Search software team composition

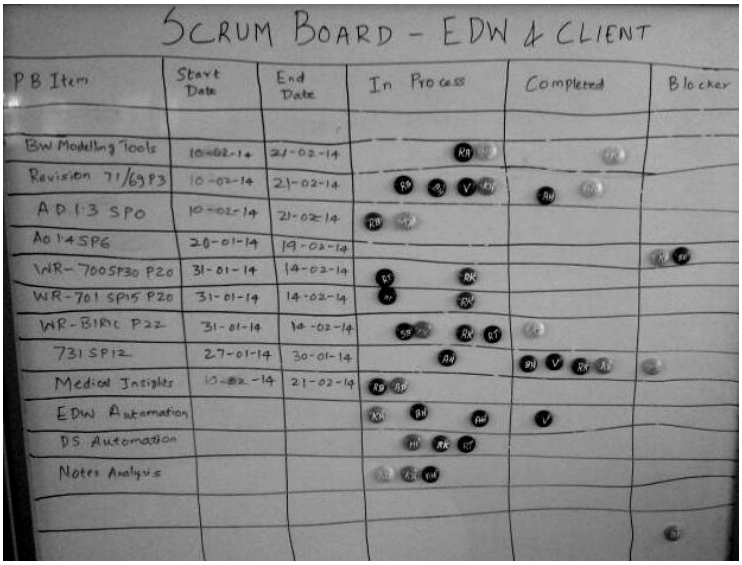

Figure 6: Scrum Board

Tables

Table 1: Review and categorization of literature on LSD

Theme of LT	Authors & Year	Results
Value and waste	Poppendieck & Poppendieck (2006)	Discussed the seven wastes in software development
	Mandic et al. (2010)	Took opposite view of Poppendieck & Poppendieck (2006) by interpreting LT from the software development angle and explained the nature of flows in software development.
	Kundu et al. (2011)	Studied twelve IT support service lines to identify waste/non-value added activities
	Kupiainen et al. (2015)	Reviews the literature to understand the metrics used in industrial Agile software development. Metrics used are focused on sprint planning, progress tracking, software quality measurement, fixing software process problems, and motivating people. Most influential metrics in studies reviewed were found to be Velocity and Effort estimate.
Lean Practices in SDP	Curtis (2011)	Jidoka was argued to be capable of detecting and eliminating defects and rework in application development and maintenance projects.
	Petersen and Wohlin (2011)	Continuous flow would help in immediately delivering the value to the software customer.
	Corona and Pani (2012)	Kanban was used in LSD to schedule work.
	Ikonen et al. (2010)	Discusses the effectiveness of Kanban for visualizing and organizing the current work
	Wang et al. (2012)	Examined 30 experience reports on applying lean approaches in agile software development published in past agile software conferences. Six types of lean application were found in the reports - non-purposeful combination of agile and lean, agile within lean out-reach, lean facilitating agile adoption, lean within agile, from agile to lean, and synchronizing agile and lean.
	Nord et al. (2012)	Documented experiences regarding the role architecture plays in lean software management practices. Release plans giving emphasis to both architecturally significant tasks and feature-based high-priority functionality will achieve improved outcomes.
	Pernstal et al. (2013)	Reviewed 38 papers on LSD, of which 42% were on large-scale development. The review concluded that research in the much-hyped field of LSD is in its nascent state in large scale development.
	Janes and Succi (2014)	Described different ideas on how to apply Lean principles within software development to answer the question how to create a “Lean Software Development (LSD)” methodology.
Specialized tools and techniques for LSD	Raman (1998)	Explored the feasibility of applying the principles of LT to software development and inferred that it is highly feasible. Study also suggested various tools and techniques such as reusability, rapid prototyping, object-oriented technologies, component-based software development, concurrent engineering, quality function deployment, etc.
	Petersen and Wohlin (2010)	Discussed a new approach in the name of “Software Process Improvement through the Lean Measurement (SPI-LEAM)”.
	Mohan et al. (2010)	SAP implementation projects use Accelerated SAP methodology similar to SDLC in software projects.
	Bastarrica et al.	Observed that software companies define and formalize their processes to make them predictable.

	(2011)	
	Alegría et al. (2011)	Designed "Analysis and VIsualization for Software Process Assessment (AVISPA)" tool to identify error patterns in different phases of LSD
	Kuusela and Koivuluoma (2011)	Proposed a lean transformation framework by focussing on the significance of learning, iterative execution, and a holistic approach
	Ahmad et al. (2013)	Analyzed the current trend of Kanban usage in software development using systematic literature review and lists the benefits obtained and challenges involved. Benefits of using Kanban were reported to be improved lead time to deliver software, improved quality of software, improved communication and coordination, increased consistency of delivery, and decreased customer reported defects. Challenges reported were lack of knowledge and specialized training.
	Bosch et al. (2013)	With an objective to provide operational support for software startup companies, this study proposes the 'Early Stage Software Startup Development Model' (ESSSDM) which extends already existing lean principles.

Source: Adapted from Anand et al. (2014) and updated by authors.

Table 2: Review of case studies in LSD

Nature of Case Studies	Author	Year	Nature of Company	Remarks
Practices, principles, and performance measures	Middleton	2001	Software Development	Performed the "before" and "after" analysis and confirmed that LSD can produce rapid quality and productivity gains.
	Middleton and Joyce	2011	Software Development	Examined how the lean ideas behind the TPS can be applied to software project management using a case study of a nine-person software development team employed by BBC Worldwide based in London
	Middleton et al.	2005	Software Development	Noted that Timberline Software in Oregon in 2002 with 450 staff was the first recorded full industrial implementation of LSD. Timberline, Inc started their lean initiative in Spring 2001 and this study recorded their journey, results and lessons learned up to Fall 2003 and thereby demonstrated that lean thinking can work successfully for software developers.
	Cawley et al.	2011	Medical Device Company	Reported about a case study of a large US medical device company, which utilized the concepts of lean principles for developing software during the design of medical devices.
	Widman et al.	2010	IMVU Inc. - a virtual worlds company	Explained the implementation of 5 traditional tenets of lean and thereby identified and reduced common wastes in software development process.
	Petersen	2012	Software Maintenance Process	Proposed four lean indicators aiming at detecting the waste in the software maintenance process using a case study of Ericsson AB.
	Rudolf and Paulisch	2010	Siemens business unit	Described how lean approaches should be interpreted for the creation of software-based systems through a case study at a Siemens business unit.
	Iberle	2010	HP Inkjet and LaserJet businesses.	Highlighted that lean is capable of handling situations which are difficult to handle using the most commonly known agile methods. Discussed implementation of lean integration in HP printer business which made complex programs easier to manage by providing visibility into what the product can and cannot do at any point in the development, improved the customer's experience by making customer workflows functional and visible early and often throughout the lifecycle, and reduced cost by driving synchronization of delivery across technology components.
	Bocock and Martin	2011	Open source project titled 'Apollo'	Studied how a high-performing, open source team adopted LSD and found that the existing meritocratic decision-making culture of the team assisted in the application of LSD. Using a case study methodology, explored how industry practitioners are using Lean principles and practices on software development projects through interviews.
	Rodriguez et al.	2013	Ericsson R&D Finland	Explored the implementation of lean principles in software development companies and the challenges that arise when applying LSD by conducting a case study at Ericsson R&D Finland.
	Staron et al.	2012	Ericsson in Sweden	Presented the *release readiness* indicator that can predict the time in weeks to release the product by studying a large LSD project at Ericsson in Sweden.
	Rodriguez et al.	2014	Elektrobit Wireless Segment	Studies how lean can be combined with Agile methods to enhance software development processes. Studies a case company named Elektrobit Wireless Segment, which has used Agile from 2007 and began to adopt Lean

Nature of Case Studies	Author	Year	Nature of Company	Remarks
				in 2010. Scaling flexibility, business management involvement, and waste reduction were found as challenges, whilst setting up teams, self-organization and empowerment appeared easier to achieve.
	Antinyan et al.	2014	Ericsson AB and Volvo Group Truck Technology	Presented a method to identify the risky areas and assess the risk involved in developing software code in Lean/Agile environment by conducting an action research project in two large companies, Ericsson AB and Volvo Group Truck Technology. Complexity and revisions metric of a source code file help in assessing its risk.
	Anand et al.	2014	Software Development	Demonstrated the application of VSM and identifies various associated waste. Proposes different lean tools to re-engineer the business process of an Indian software firm that provides supply chain software solutions to logistics providers.
	Kirovska and Koceski	2015	Software development in a IT company	Presented Kanban methodology and its practical usage within a software development environment. Practical implementation of this concept is presented using a web-based application called KanbanMAK within an IT company.
Organization and human resources requirements for LSD	Norrmalm	2011	SDP of a large manufacturing-oriented organization	Improvement areas in terms of lead time and quality were identified using VSM and a framework of seven common improvement areas in software development was designed.
	Malladi et al.	2011	IT service industry	Identified some of the best practices in lean methodology as applicable to the IT service delivery and used a case study approach to demonstrate its application.
	Staats et al.	2011	Wipro(Indian software services firm)	Examined the applicability of lean production to knowledge work and found that LSD projects performed better than non-lean projects.
	Ikonen et al.	2011	R& D Software Company	Studied the impact of Kanban on software project work and concluded that it provided considerable benefits in the form of motivation and control over the project activities.
	Gong & Janssen	2014	IT service organization and its two IT outsourcing providers	Developed a conceptual framework to describe how Lean can drive IT service innovation within IT outsourcing relationships. A clear strategic direction and learning environment were found to be critical for achieving IT service innovation.
	Mäntylä et al.	2015	Mozilla Firefox	Investigates the changes in software testing effort after moving to rapid releases in the context of a case study on Mozilla Firefox. Rapid releases have a narrower test scope that enables a deeper investigation of the features and regressions with the highest risk and it makes testing more continuous with proportionally smaller spikes before the main release.

Source: Adapted from Anand et al. (2014) and updated by authors.

Table 3: Problems mapped to wastes and LSD practices implemented as solutions in the case organization

S. No.	Current Problems	Related Wastes	LSD Practices as Solutions	
1	Lack of intra-team integration: Updates were shared within the team less frequently	1) Over-processing 2) Defects 3) Relearning 4) Reworking	1) Fast Feedback: Daily Scrum meetings and sprint review 2) Improved synchronization (Andon/Jidoka) within the team 3) Concurrent software development	4) Documentation of repetitive tasks into modules for direct use 5) Documentation of individual's tasks for future knowledge transfer
2	Lack of cross team integration: Updates were shared across the teams rarely	1) Over-processing 2) Waiting 3) Defects 4) Relearning 5) Reworking	1) Regular meetings of teams working on similar projects: Sprint retrospective 2) Improved synchronization (Andon/Jidoka) across the team.	3) Cross team knowledge sharing through employee involvement 4) Access to tacit knowledge
3	Long release cycles, late shipments, high lead time and high waiting time: Time consumed were mostly greater than the targeted timeline	1) Partially done work 2) Waiting 3) Motion 4) Task switching 5) Handoffs, Defects & reworks 6) Unrequired extra features (Over-processing) and lacking required features	1) Submitting the software products in as it is state on target date 2) Incremental development with in-between customer interactions (defect prevention through self-inspection and successive inspection)	3) Regular feedback 4) Conceptual integrity between customers and developers 5) Pull system
4	High operations cost: Actual cost exceeded the budgeted cost	1) Waiting 2) Task switching 3) Defects 4) Handoffs 5) Redundant training 6) Rework	1) Fast Feedback 2) Concurrent software development	3) Increasing expertise through knowledge sharing
5	Defects, bugs, errors leading to rework in the final product	1) Partially done work 2) Waiting 3) Motion 4) Defects 5) Rework	1) Fast Feedback from customer 2) Pull system	3) Reduced repetitive iterations 4) Improved synchronization (Andon/Jidoka)
6	Late changes in software specifications	1) Partially done work 2) Waiting 3) Motion 4) Defects 5) Rework	1) Fast Feedback 2) Conceptual integrity between customer and developer	3) Pull system 4) Improved synchronization (Andon/Jidoka)
7	Lack of individual team member involvement	1) Defects 2) Relearning 3) Waiting 4) Rework	1) Regular meetings to listen to their issues and problems (both technical and personal)	2) Employee involvement 3) Suggestion schemes 4) Expertise increased
8	Unclear requirements & responsibilities, inefficient scheduling, and information	1) Partially done work 2) Waiting 3) Defects 4) Rework 5) Motion	1) Regular meetings varying from daily to monthly at different levels in the organization 2) Clarity on what to be done at the day level, month level, and	3) Meetings to explain the contribution of each individual task to the complete final product (to attain global optima than optimizing at individual levels)

	asymmetry		task level.	4) Conceptual integrity
9	Lack of standardization of repetitive task	1) Waiting 2) Relearning and reworking 3) Extra features 4) Extra processes 5) Excess processes	1) Modular development: Standardized modules for repetitive tasks	2) Value stream improvement: Improving on value added tasks and reducing non-value added redundant tasks

Table 4: Performance measures before and after LSD adoption

Before LSD	After LSD	% Improvement
Frequency of within the team meeting = 5 per month	Frequency of within the team meeting = 11 per month	120% increase
Number of instances of knowledge transfer within the team = 2 per month	Number of instances of knowledge transfer within the team = 10 per month	400% increase
Frequency of cross team meeting = 0 per month	Frequency of cross team meeting = 1 per month	100% increase
Number of instances of knowledge transfer across the team = 0 per month	Number of instances of knowledge transfer across the team = 1 per month	100% increase
Average release cycles (lead time) = 6 months	Average release cycles (lead time) = 1 month	83% decrease
Percentage of late shipments = 10%	Percentage of late shipments = 2% (only when high priority bugs identified)	8% decrease
Average delay in shipments = 5 days	Average delay in shipments = 1 day	80% decrease
Average intra-team waiting time = 7 days	Average intra-team waiting time = 1 days	86% decrease
Total number of issues raised = 165	Total number of issues raised = 15	91% decrease
Actual number of valid defects (bugs) = 66	Actual number of valid defects (bugs) = 12	82% decrease
Bug ratio (actual number of accepted defects by total number of issues raised in the end product) = 40%	Bug ratio (actual number of accepted defects by total number of issues raised in the end product) = 80%	40% increase
Percentage of major reworks (when time taken to rework is more than 15% of the time taken to develop it) = 40%	Percentage of major reworks (when time taken to rework is more than 15% of the time taken to develop it) =15%	25% decrease
Percentage of minor reworks (when time taken to rework is less than 15% of the time taken to develop it) = 60%	Percentage of minor reworks (when time taken to rework is less than 15% of the time taken to develop it) = 35%	25% decrease
Average number of e-mails transacted = 180 per month	Average number of e-mails transacted = 320 per month	78% increase
Average number of suggestion schemes by a team = 3 per month	Average number of suggestion schemes by a team = 6 per month	100% increase
Average number of tickets raised per team= 13 per month	Average number of tickets raised per team = 25 per month	92% increase
Percentage of task switching = 30%	Percentage of task switching = 7%	23% decrease
Number of days dedicated to technical training = 14 days in 6 months	Number of days dedicated to technical training =7 days in 6 months	50% decrease