We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

3,500
Open access books available

108,000
International authors and editors

1.7 M
Downloads

Our authors are among the

151
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

BOOK CITATION INDEX
CLARIVATE ANALYTICS
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

A blended process model for Agile software development with lean concept

Indika Perera
University of Moratuwa
Sri Lanka

1. Introduction

This research addresses a known set of issues with Agile Software Development through a unique form of solution. In fact, the research approach can be considered as one of the first ever research to propose a hybrid process paradigm to overcome Agile process issues with the assistance of Lean manufacturing principles. Research results show a significant improvement for the normal Agile practices, which indeed a unique and worthy finding for Agile practitioners. After years of being practiced in the industry the Agile software development process possesses standard characteristics of a process paradigm (Perera, 2009). However, due to the inherited higher degree of flexibility and the exceptional abstract nature of the process principles, Agile process heavily depends upon the project and people norms once it is implemented. Having more flexibility is a better attribute for a process, if it is used by competent experts who can take productive decisions at right moments. However, depending too much on expert knowledge to process and product adjustments is a questionable concern to a growing project with rapid changes to its development and releases.

Software applications are complex and intangible products, which are difficult to manage. Hence Software Lifecycle management becomes one of the key research areas in software engineering. Due to the nature of the software, software researchers and practitioners are focused on improving the software processes which are used to develop software. The underline assumption is that there is a direct correlation between the quality of the process and the quality of the developed software (Fuggetta, 2000). A software process can be considered as a set of tools, methods and practices, we use to produce a software product (Humphrey, 2006). These are the prime parameters, also known as Key Process Areas (KPAs), that differentiate the process based software development from ad-hoc programming. Identifying KPAs is one of the main considerations when a certain process model to be improved (Fatina, 2005). In this research, the KPAs of Agile practice were studied and reviewed for required improvements, considering criticisms on those. Specially, the driving KPAs of Agile practice such as, the non-standardized process flow, reliance on key people, and immense flexibility were the considerations for this study. Then the Lean principle for possible key practices to incorporate with classical Agile practice was examined.

The remainder of this chapter is arranged into 8 sections as follows: section 2 provides some background literature on the major areas of interest with respect to this research; section 3 describes the research problem this research worked on as an extension to the literature review. The section 4 presents the proposed blended process model as a solution to the research problem being considered. Section 5 and section 6 elaborate the detail on the experiment conducted to evaluate the proposed process model and the analysis of the results obtained, respectively. The Section 6 describes possible policy implications and future work before concluding. Finally, the section 8 with references completes the chapter.

2. Background

This section includes a comprehensive synopsis of the literature referred for the study. In fact, the main emphasis was given on the topics; the Agile software development, the Lean principle, and the Lean software development. Therefore, this section is divided into three main areas of literature, representing the focus of the study. There is a plethora of case studies and application stories on Agile software practice and Lean principle in an isolated manner. As the paper explains in the problem section, most of those cases do not emphasize the possible improvements to the two practices to overcome their weaknesses. Further, there are concerns of using these two practices in certain applications and specific cases, considering their weaknesses. For this study, it was considered that the proposed blended process model should be derived upon the fundamental parameters of the two practices, for the simplicity and to obtain a generic process model as the outcome. Hence, the literature focus was decided to be on fundamental concepts of the selected two practices than their applications or customized models.

2.1 Agile Software development

Agile software process was introduced and defined by a group of experts in a collective nature, to overcome issues with the traditional software processes. Agile Manifesto was the proper introduction of the Agile methods to the software industry. According to the Agile Manifesto the following four norms are the basics of the Agile methods.
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan (Agile Manifesto, 2001).

Most of the traditional software processes suffer from having heaps of documents once the project finishes. Despite from those most obvious differences between plan-driven life-cycle models and Agile development is that Agile models are fewer documents oriented and place more emphasis on code development (Perera & Fernando, 2007). By the nature of this paradigm, it also provides some other benefits like, flexible project management, cost effective adaptability (Perera & Fernando, 2009), increase communication and ultimately increased customer satisfaction (Perera & Fernando, 2007). Agile Methods are a reaction to traditional ways of developing software and acknowledge the need for an alternative to documentation driven, heavyweight software development processes (Cohen, *et al.*, 2003). Augustine (2005) has defined Agile Software Development as the work of energizing, empowering, and enabling project teams to rapidly and reliably deliver business value by

engaging customers and continuously learning and adapting to their changing needs and environments.

Agile software development which emphasizes sense-and-respond, self-organization, cross-functional teams, and continuous adaptation, has been adopted by an increasing number of organizations to improve their software development (Lee and Xia, 2010). However, it was observed that when applied to large scale industrial projects, Agile practices fail to keep their stability and performance measures within the expected norms (Perera & Fernando, 2007). This also confirms the fact of the unbalanced number of many successful Agile projects teams with few developers, ideally less than 10 persons. Agile techniques have demonstrated immense potential for developing more effective, higher-quality software. However, scaling these techniques to the enterprise presents many challenges (Shalloway, *at el.*, 2009). One of the main objectives of this study to raise the stability of Agile process with Lean principle, which may help to sustain with large development teams, even though the experiment environment of this research does not incorporate such teams. Moreover, in the Agile world, requirements change rapidly developers expect this and are not fazed by the possibility of having to discard their work and start over (Black, *at el.*, 2009). However, the software process and productivity standards and norms believe that such level of work discard and alterations are essentially impact to the end productivity; more or less it will be compensated either by compromising the customer expectations or more frequently, at the expense of the developer time. This factor was considered as a prime motive when the experiments were designed to assess the productivity of the proposed process model. More detailed analysis on Agile process issues is included in the section 3.

2.2 Lean Principle

The Lean principle has a long history of application in Japanese automobile industry, especially within the Toyota manufacturing process. Taiichi Ohno has done a pioneering work to introduce the Toyota Production System with based on Lean concept. Taiichi Ohno and Shigeo Shingo introduced their new concept to the Toyota Production System in result in a significant productivity boost in early 1950 (Ohno, 1988). After few decades of the Lean concept introduction in Japan, many researchers around the world began to investigate possible applications of this concept as a generic production model. The early articles were named as Toyota Production System instead of the name Lean concept/principle, and the first English article was published by Sugimori, *et al.* (1977) on the principles of the Toyota Production System. However, with the book on 'Lean Thinking' by Womack and Jones in 1996 has triggered the momentum on Lean applications to various industries and relevant researches (Womack and Jones, 2003). More interestingly, software development (Poppendieck, 2007) and pharmaceutical (Petrillo, 2007) industries were the early adapters of this new manufacturing concept, spanning across two segments of the commercial arena; the service sector and manufacturing sector, respectively. More discussion on the Lean Software Development is included in the next section.

The Lean principle is based on two practices: the elimination of the waste (Muda) from the production process, and the continuous quality inspection, known as Jidoka, within the production process (Danovaro, *at. el*, 2008). In fact, Jidoka is an operational process which ensures the waste is within the expected amounts or at zero levels. Therefore, essentially, the elimination of waste is the fundamental objective of the Lean principle, although, there are derived and extended applications can be seen, to date.

There are five basic principles of Lean manufacturing: as Specify value, Identify all the steps in the value stream, Flow smoothly, Pull value, and Pursue perfection are the five principles in Lean practice (Womack & Jones, 2003).
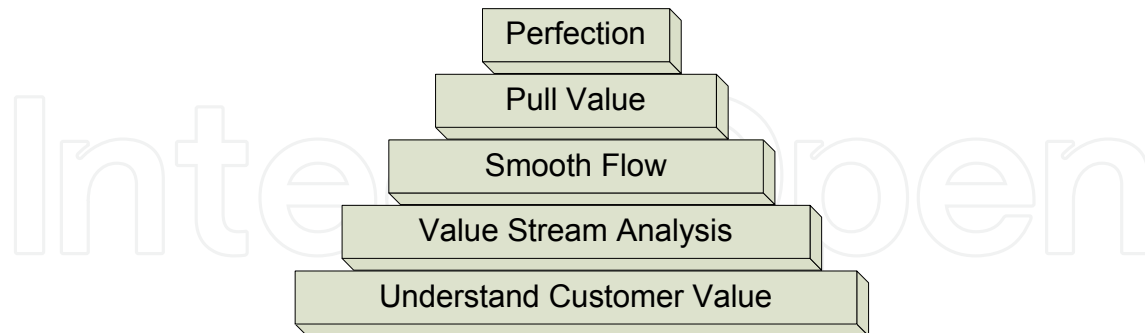

Fig. 1. Lean Principle Model – Five Basic Principles and Their Relationship

Step 1 - Understand Customer Value—Value must be externally focused. Only what customers perceive as value is important for the development.

Step 2 - Value Stream Analysis— Once the value that is required to deliver to the customers has been identified, you need to analyze all the steps in your business processes to determine which ones actually add value. If an action does not add value, you should consider changing it or removing it from the process.

Step 3 – Smooth Flow—Instead of moving the product from one work centre to the next in large batches, production should flow continuously from raw materials to finished goods in dedicated production cells.

Step 4 – Pull Value —Rather than building goods to stock, customer demand pulls finished goods through the system. Work is not performed unless the part is required downstream.

Step 5 - Perfection—As you eliminate waste from your processes and flow product continuously according to the demands of your customers, you will realize that there is no end to reducing time, cost, space, mistakes, and effort (Womack and Jones, 2003).

Lean principle is composite with unique methodologies to perform the operational activities. Kanban (Pull) production system is one important method (Gross, 2003). In that approach, throughout the production lines one can schedule the value flow process efficiently, and activities flow using signalling to each other with respect to the workflow. In 1953 Toyota applied this logic in their main plant machine shop (Ohno, 1988). Just In Time (JIT) is the basic process Toyota used, and the Kanban is an improved process of the JIT (Kupanhy, 1995). For this research Kanban was identified as a key element of the proposed blended process model to facilitate value flow without an overhead burden to the Agile software practitioner.

2.3 Lean Software development

Applying Lean principles to software development projects has been advocated for over ten years, and it will be shown that the extensive Lean literature is a valuable source of ideas for software development (Middleton, *at el.*, 2007). One of the domains affected by the Lean Thinking was the Software Development, which generated the term Lean Software Development (Udo, *at el.*, 2008). The first glimpse of Lean Software Development was appeared with the research work done by Middleton (2001), on two industry case studies of

software engineering with Lean implementation. However, Mary Poppendiek and Tom Poppendiek (2003) were the pioneers of introducing a more enhanced software development practice based on Lean principle, which was branded as Lean Software Development.

Lean Software Development mainly focuses on defect minimization within the software development activities. Effectively, the Lean Software Development model has been able to map seven wastes in production systems to software domains; a brief overview of this mapping is shown in the following table 1. It is adapted from the work of Poppendiek & Poppendiek (2003), and Poppendiek (2007).

Wastes in Production Domain	Corresponding wastes in Software Domain
Overproduction	Extra Features
Inventory	Requirements
Extra processing steps	Extra steps
Motion	Finding Information
Defects	Defects not caught by tests
Waiting	Waiting, including customers
Transportation	Handoffs

Table 1. Corresponding seven types of waste in software development

There are some criticisms on this way of thinking with the software development activities. Specially, even though these seven waste types and the Lean Software Development practices are successful enough to minimize defects of the software development, this abstract way of process mapping does not provide a sufficient level of information for a comprehensive software process practice. In deed that is the key issue with using Lean Software Development practice in large scale industry projects. In fact, Lean Software Development does not incorporate the key software lifecycle activities, such as Requirement Engineering, Software Design, Software Testing and Software Deployment, but the Software Development. Without any of these key steps it is rather inappropriate to consider Lean Software Development as a software process model for generic use.

3. The Research Problem

The main purpose of this research was to formulate a new software process paradigm model and evaluate its success. Having said so, let's consider the research problem that this research tries to address. In fact, this research mainly considers Agile practice and Lean concept, as the research's basis. Agile development has significantly impacted industrial software development practices; though it's wide popularity, there's an increasing perplexity on software architecture's role and Agile approaches (Abrahamsson, at el., 2010). The team lead engineers and the software development managers may be unsure how to adopt Agile methods incrementally, which situational practices should perform, and how to engender enthusiasm in team members (Syed-Abdullah, et al., 2007). Chow and Cao (2008)

have done a survey study to identify critical success factors in Agile software projects which they have categorized into four major aspects; Quality, Scope, Time, and Cost. This also indicates that precise settings for quality, scope, time and cost will result in a successful Agile software project. The Agile development literature is largely anecdotal and prescriptive, lacking empirical evidence and theoretical foundation to support the principles and practices of Agile development (Lee and Xia, 2010); this also indicates that there is a concern on standard practice of Agile principles and norms across the industry. Mainly, due to the high flexibility and lack of awareness, Agile practitioners interpret their own forms of Agility, where in most of the cases deviate heavily from the optimum Agile best practices. In terms of scalability Agile practices are usually applied to projects with smaller teams of ten or fewer people (Deek, *at el.*, 2005). This limitation is also considered due to the uncertain and highly expert based interpretations and implementations of the basic Agile principles.

There have been many efforts to improve Agile practices but to date some of the Agile process based software projects suffer due to the weaknesses inclusive to the process practices and individual forms of Agile applications. However, there is no successful method to address the behavioural issues with Agile practices and standardizing it for a uniform practice independent from expert judgment, unfortunately. Process improvement offers a sustainable method of making project success probability a significantly higher value irrespective of individual dependencies (Jacobs, 2006). Salo and Abrahamsson (2005) have done an empirical study on Agile software development integration with software process improvement, where they state continuous improvement of Agile software development processes is important in enhancing the capabilities of the project members and the organization as a whole. Miller and Sy (2009) have indicated an extensive summary of factors that affects Agile software processes, where the major concerns are concentrated on aspects such as poor communication, lack of expertise for autonomous development, weak value flow, dependency issues, etc. Essentially, this provides an excellent arena to perform Lean practices along with Agile process as a remedy; The Lean principle more or less address most of these issues in a more flexible manner where agility could not be successful. Lean manufacturing has a proven set of records for flexible and productive manufacturing in many industries. However, Leanness alone may not be appropriate for software development, instead of agility, as the basic Lean model focuses on defect minimization as the prime objective. Narasimhan and others (2006) have indicated that Agile practice could presume leanness but leanness might not presume Agile nature. This is an interesting claim. However, there are no significant further studies done afterwards.

Therefore, as the basic problem domain of this research, the above mentioned Agile process weaknesses have been considered. The research has successfully tried to formulate a blended process model with combining appropriate Lean principles with the Agile software process to improve the Agile process stability, certainty, and productivity without compromising its advantages.

4. The Blended Process: Lean-Agile hybrid model

Yusuf and Adeleye (2002) have compared the effectiveness of Lean and Agile manufacturing in UK. Even though, the conclusions were derived that Agile manufacturing slightly outperform Lean manufacturing, there is no comprehensive study have been done

on software development context. Further, the research focus on agility and Lean practices in software development have been more or less equally supportive observations for both Agile and Lean software development approaches. Therefore, this research was driven with the prime motivation of combining the best aspects of the both processes to form a blended process model.

Santana (*at el.,* 2009) have indicated that the focus of Agile project learning should be on improving the performance of the ongoing project. This continuous learning with an ongoing project will effectively increase the quality and productivity of the produce being developed. It is important to have a parallel vigilance over the Agile activities, as they are more or less implemented according to the individual project and people norms. Prince and Kay (2003) combined Agile manufacturing with Lean concept to achieve better production flow. Integrating Lean and Agile characteristics becomes an important study on how these philosophies can assist business to prosper (Naylor, *at el.,* 1999), although, software process development researches have not been guided to a reasonable extent so far, unfortunately. The solution for the Agile process poor scalability is to integrate the principles and practices of Lean with Agile ideology and methods (Shalloway, *at el.,* 2009). Moreover, to combine Agile process with Lean practices, it is required to ensure that there will not be redundant process steps in the outcome due to the higher degree of similarity between the two processes being considered. Though Agile and Lean practices are appeared to be similar, there are basic differences between the two in the context they are applied; the difference is in the underlying perspective and philosophy and the mindset (Hibbs, *at el.,* 2009).

As briefly explain in the section 3 above, with this blended process model, the identified key Agile weaknesses were addressed through Lean practices. In that sense, the prime objective of this proposed model was to increase the process stability, developer autonomy, and higher degree of productivity over the classical Agile practices. It was hypothesised that incorporating Lean streamlined routine based activities within Agile development phase would help to achieve these objectives. Specially, the main hypothesis was that through Lean incorporation, developers get more time to focus on their development work than worrying about the process management activities.

4.1 Lean-Agile Hybrid Model

As the first step towards the model development, a simple value stream map was developed respective to the Agile process based on the basic classical Agile principles and standard practices. According to Oppenheim (2004), the current-state of the value stream map enables the identification of wastes and possible improvements; hence, using this value stream, possible weak spots were identified comparing the Lean value stream against the Agile. Even though this was a trivial activity, it was one of the crucial steps for the success of the model. One of the major drawbacks with classical Agile value stream map was the higher degree of developer effort to streamline the development process. Literary, this is an overhead task for a developer. Unfortunately, the role of the project manager is not significant at the grass root level of development in an Agile team; hence development team members should perform relevant scheduling and workflow management, individually. This can also affect to their decision making on the technical designs as well. In the proposed model, these possible overhead points were incorporated with relevant Lean steps at the micro level. The underline hypothesis was to inject routine practises of Lean steps into

flexible yet weak Agile process points where the blended process will have more certain and stable process points over classical Agile, respectively.

With this understanding of the Agile process weak points, the proposed blended process model was developed with a 4 step Lean model; in fact, one step of the altered Lean model is a combination of the basic steps 'Smooth Flow' and 'Pull Value'. It was identified that these two core Lean principles can be combined and practiced along with the Agile Development phase. Literally, the Development phase of the Agile practice overwhelms significantly the other phases; it further justifies the decision taken to merge these two Lean steps. The proposed model and the classical Agile model are shown in the following figure 2.
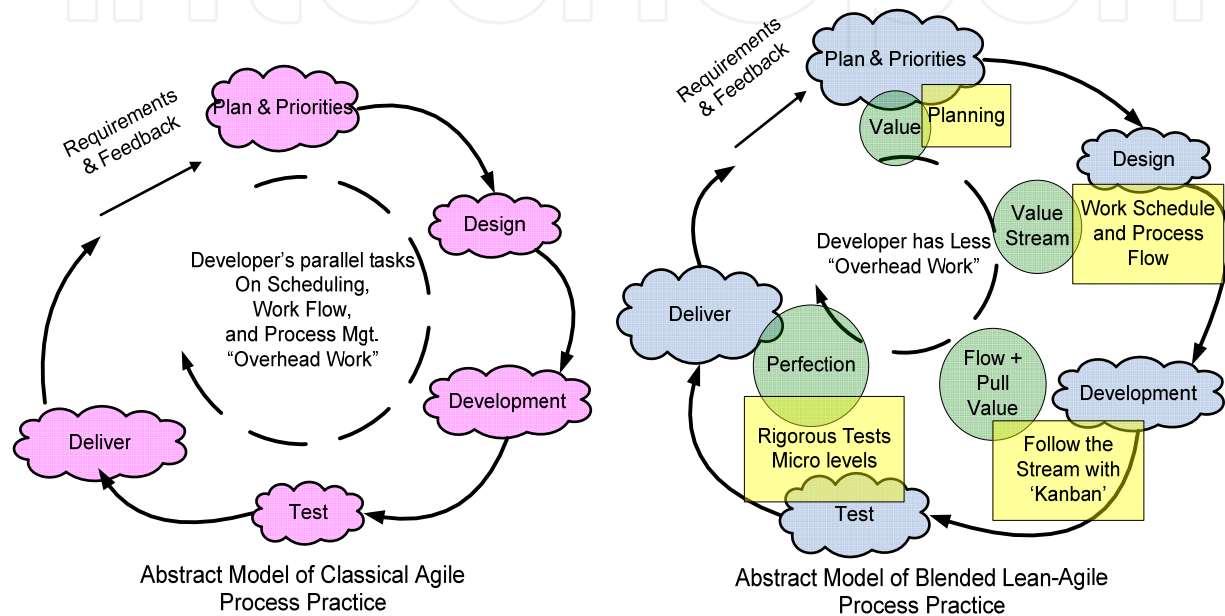
Fig. 2. The Classical Agile Process Model (Left) and the Proposed Lean-Agile Blended Process Model (Right)

The proposed model mainly tries to address the issue of overhead work on an Agile developer, despite the expected Development work. Specially, this is a significant issue which is the fundamental cause for many Agile weakness described above. With the value stream map, it was identified that due to the nature of Process Micro Management by the individual developer this overhead work can be significant, and affects the developer productivity in a substantial manner. With the incorporation of Lean steps the process expects to routines most of the trivial work parallel with the development work, without much effort from the developer end. However, the mere incorporation of the altered Lean steps with relevant Agile activities would not give a practicable model with realistic steps. Therefore, a further step towards process implementation was incorporated in a more tangible manner. The rectangular boxes in the figure represent these steps which ware the linkages with abstract Lean principles and Agile phases as appropriately. The first step – Planning represents the initial action towards the respective iteration of the Agile process, where it covers the value understanding and development priorities for the iteration. The second step – the work schedule and process flow is the effective starting point of the proposed model. There the developers decide how their development work (Value Stream)

should be, and they decide the process flow. The most significant improvement can be seen with third step – following stream with 'Kanban', where each developer (or pair) should follow the decided the value stream based work through 'Kanban' cards. A Kanban card is a simple form of signalling mechanism between the workers of a Lean practice. Anybody can define their own Kanban cards according to their requirements. Though it is a simple technique, it has a proven record of success. Specially, a person who follows the process with Kanban does not have to think about the process at all, but the work assigned with that Kanban card. A simple, yet sufficient Kanban card was designed for the experiments. One of the used Kanban cards is shown in the following figure 3.

Class/ Interface/ Item	Status	Location/Path	
Table ViewSummary	urgent	/root/clientGUI/	
Description: To show Processing Summary on the Client Panel		Order Date	01/12
		Due Date	01/14
Request By	Pair 2	No: 0034-1	
	Gharith	Card 1 of 2	

Fig. 3. A Snapshot of a Used Kanban Card during the Experiment

As the final step of the proposed model, a rigorous testing at the micro level was introduced as a perfection norm. This testing effectively higher granular than unit testing, making lesser load on unit testing and sub system testing activities, while giving more opportunities to find errors in the code, specially before they are being camouflaged. Beyond this step, the blended process reiterates with the next cycle of the development similar to the classical Agile process.

5. The Experiment Methodology

Conducting an experiment to evaluate a software process is not an easy task. There are various study types that can be performed. In many cases, the type of study will depend on the circumstances. Much of what we do in the software engineering domain is opportunistic, and we are often limited by the situation available (Basili, 2007). However, "The approaches vary in cost, level of confidence in the results, insights gained, and the balance between quantitative and qualitative research methods" (Basili, 1993). First, the experiment methodology should comply with the objectives of the underlined study. Further, the experimental paradigms require an experimental design, observation, data collection and validation on the process or product being studied (Basili, 1993). With these objectives in mind, following experiment environment was designed to evaluate the proposed process model's success over classical Agile process.

5.1 The Experiment Environment

This experiment was designed to evaluate the introduced new hybrid software process paradigm's appropriateness in the practical environments. The experiment environment and the performance measures were carefully selected in accordance with the prime objective of the experiment and to suit with the research hypothesis.

The experiment environment was selected in a way to practice the two software paradigms, collect their respective measures, and analyze the collected results. Therefore, it was decided to conduct the experiment with a controlled sample.

The final year student projects of the Department of Computer Science and Engineering (CSE), University of Moratuwa, were the best possible test samples available for this study. Those final year projects created a homogeneous experiment platform among each project. For this study, 10 final year project groups were selected to participate in this experiment in voluntarily basis. Each project group was consisted of 4 final year CSE students.

Before the experiment 4 mini workshops were conducted for the entire sample (10 groups) on the Agile practice of software development. This was to ensure to diminish the knowledge gap on the Agile process between the groups as well as between the group members within a group. After that 5 groups were separated from the rest to follow the new Lean-Agile blended process model, which is considered as the experiment sample. These 5 groups were selected entirely upon the voluntarily basis. For this experiment sample, 3 additional mini workshops were conducted to familiarize the Lean principle and the new process model. However, extra measures were taken on planning and delivering of these 3 mini workshops to ensure no additional Agile process skills were developed on the experiment sample students over the controlled sample. The other five groups were asked to follow the classical Agile practice, which was considered as the controlled experiment sample. While leaving both samples a fortnight to practice their process methods, the experiment phase started. Data gathering was done thereafter for 10 weeks, on per group basis, within the Software Development phase of their projects. A typical working instance model of a group which practiced the proposed blended process is shown in the figure 4 below.
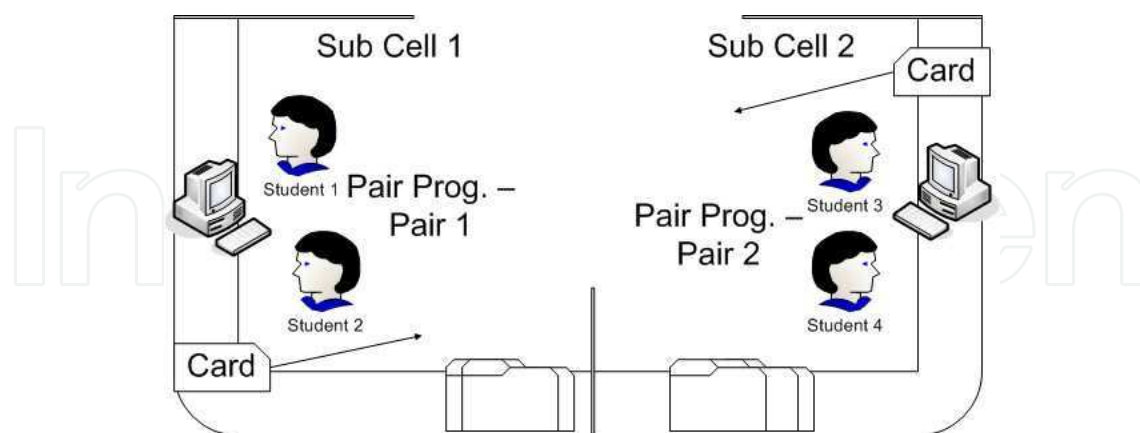

Fig. 4. The Lean-Agile Value Cell – An Instance of a Student Group Development Work

5.2 Performance measures

Since the experiment was focused on comparing the two samples to find out which one is better with respect to the software development, a set of essential measurement parameters

were identified. One important fact to mention here is that the division of the Lines Of Codes (LOC) parameter to three parts as New LOC, Changed LOC, and Removed LOC. If only total LOC is measured, changes to where the LOC comes from may go undetected (Rozum, 1991). These changes may reflect significant differences in the effort and schedule required to complete the project. Performance measures are shown in the following table 2.

Measurement	Amount	Work level (Human hours)
New LOC	N_LOC	H_NLOC
Changed LOC	C_LOC	H_CLOC
Removed LOC	R_LOC	H_RLOC
Defects fixed	D	H_D
Expected work	--	H_EW
Actual work	--	H_AW

Table 2. Measurement Parameters for the Experiment

The number of defects fixed is important to understand the difference between the two practices in the context of quality enhancements to the developed software. Expected work amount and the actual work amount values were used to compare the effective work completion rate between two paradigms.

All these performance attributes were measured as quantitative values during the examined time period along with their respective work amount in human hours. This human hour measure is very important to understand the difference of work loads in the classical Agile and the improved Agile process model. In addition to that, the human hour values were used to identify the weighted factors for the statistical analysis on the three types of LOC; New, Changed and Removed, measured during the experiment. Furthermore, in some cases, human hour values were used significantly to verify the respective LOC values for their accuracy. One could question the appropriateness of the selected experiment environment and the measurement parameters for this study. However, due to the following reasons, the experiment methodology can be justified steadily.

5.3 Experiment Rationalization
Software process related experiments are heavily suffered by the people factor. Process can provide a useful framework for groups of individuals to work together, but process per se cannot overcome a lack of competency (Cockburn, 2001). Individual competency discrepancies on software development can cause varying results in the experiments. But, the selected participants of this experiment have the least skill differences when compared with the other possible participant samples. The same year (final year) students, who have followed a more or less similar set of courses and projects, can be considered as equally skilled developers, compared to generic sample of developers. Industrial software firms always have different competent software developers for their projects, and organization to organization, people competencies differ significantly. It affects to the homogeneity of the sample participants to a significant extent.

Another aspect is the scope of the selected projects. All these projects are worth of 10 GPA credits for the students' graduation. Therefore, initial guidance on project scope was given to the students. In addition to that at the beginning of the project work, a set of experts analyzed those project proposals and ensured to keep the projects within the expected scope for a final year project. If the industrial projects were taken into the experiment, this type of similar scoped projects may not be available. Therefore, it is reasonable to assume all projects have a similar work level required for their completion.

Furthermore, the final year students have equal time and resource constraints while practicing their project with their other academic activities. Specially, this was a key success factor to constrain student development work to have a uniform experiment environment.

All these create the best experiment platform one could ever find to this type of an experiment. If the samples were taken from the industry, this kind of uniformity would not be possible, since different organizations have different resource levels and different schedules for the completion of their projects. As the experiment is sensitive to relative measures, that kind of project environment can create too much deviated results from the tolerant levels.

6. Results and Analysis

6.1 Analysis – Hypothesis Testing 1

Though the main objective for this research was defined at the beginning, for this analysis a derived hypothesis from the main objective was used. In fact, what has been evaluated in this analysis was the main objective of the study, but using a slightly different hypothesis, merely because to align the analysis with the data and the objectives of the study.

In this analysis, the software development productivity rate was considered as a measure of the effective Line of Codes (LOC) being produced. With that perspective, following hypotheses were defined for the analysis.

Null hypothesis (H_0) – Agile software development productivity cannot be improved by combining Lean principles

Alternative hypothesis (H_1) – Agile software development productivity can be improved by combining Lean principles

Since the analysis is based on LOC and the collected data samples have three different LOC values, i.e. New LOC, Changed LOC, and Deleted LOC, weighted summations of those three were derived on per group, per week basis. Considering the human work hours spent on each category and the usefulness to the final code, following three weights were identified. $W_N = 1$ for New LOC, $W_C = -0.5$ for changed LOC, and $W_R = -1$ for removed LOC.

Using these weights, 50 data values were derived for a sample and the values are shown in the following tables for the two samples. For a given Week (W_i) and for a given Group (G_i), the LOC value was obtained as the equation (1).

$$LOC = W_N * N\_LOC + W_C * C\_LOC + W_R * R\_LOC \qquad (1)$$

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
G1	55.5	80.5	191.5	320	365	409.5	373	557	462	668
G3	67	107.5	145	275	375	377.5	420	585	744.5	575
G4	66	112	168	301	345	397	336.5	192	580.5	442.5
G9	52.5	187	200.5	475.5	676.5	581.5	444.5	567	745	944
G10	48	203.5	364.5	481	551.5	659	672.5	751.5	386	908.5

Table 3. Sample A (Classical Agile Process) weighted sums of LOC

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
G2	102.5	204	240	391.5	486	542	533	533.5	680.5	540.5
G5	107.5	170	375	494	574.5	470.5	220	610	501	755
G6	97	221	388.5	751	692	812	594.5	939	688.5	779
G7	130.5	380.5	570.5	695.5	844.5	919	879.5	1081	1088	699.5
G8	49	342	435	86	920.5	1097	1084	968	943	981.5

Table 4. Sample B (Lean-Agile Process) weighted sums of LOC

To compare the samples with above data and test the hypothesis, Analysis of Variance (ANOVA) statistical method was selected. Instead of manual calculation, the Minitab© statistical application was used. Minitab release 13.20 was the application version which used for this analysis. As the name implies, ANOVA is based on variance analysis between the samples, and it is a widely used statistic model for comparing two or more samples for their means. Actually, the Analysis of Variance (or **F-test**), as with Student's **t-test**, is fairly robust with respect to violations of the assumptions of normality and homogeneity of variance, so the primary claim is that of equality of means; the alternative hypothesis, then, is that at least one of the population means is different from the others. The **p-values** derived from its use are not strongly affected by such violations, as long as the violations are not too extreme (Vokey and Allen, 2002). ANOVA uses the following equation (2).

$$SST = SSW + SSB \qquad (2)$$

This is the fundamental equation of ANOVA; the unique partitioning of the total sum of squares (*SST*) into two components: the sum of squares within groups (*SSW*) plus the sum of squares between groups (*SSB*). This is a very abstract model, though the computations of those values are somewhat complex, however, further detail of ANOVA is beyond the scope of this research. The Minitab output for the ANOVA on this hypothesis test is shown in the figure 5, below.

```
Welcome to Minitab, press Fl for help.
Retrieving project from file: I:\PROGRESS2ND\MINITAB.MPJ
```

One-way ANOVA: C1, C2

```
Analysis of Variance
Source      DF        SS        MS        F        P
Factor       1    755856    755856    10.23    0.002
Error       98   7238383     73861
Total       99   7994239
                                      Individual 95% CIs For Mean
                                      Based on Pooled StDev
Level        N      Mean     StDev  --------+---------+---------+-------
C1          50     399.9     233.0  (-------*-------)
C2          50     573.8     305.6                    (------*-------)
                                    --------+---------+---------+-------
Pooled StDev =    271.8              400       500       600
```

Fig. 5. Minitab output - ANOVA output for hypothesis test on effective LOC

Sample A: The mean value μ_A = 399.9 LOC/week and the standard deviation σ_A = 233.0.
Sample B: The mean value μ_B = 573.8 LOC/week and the standard deviation σ_B = 305.6.

From the average developed LOC values for a week, clearly, the blended Agile practice is capable of producing more effective LOC than the classical Agile practice; hence, a higher productivity. However, just by considering the means of the samples, hypothesis tests cannot be done, statistically. With the ANOVA test, the p-value or the significance level was 0.002 for the groups A and B. In ANOVA, to reject the H_0 the p-value should be less than 0.05 and if not the H_0 will be accepted. In this case, the p-value is 0.002 i.e. $p < 0.05$; therefore, reject the Null hypothesis (H_0) with 95% confidence level. This implies that the Agile process development productivity can be improved by applying Lean practices.

6.2 Analysis – Hypothesis Testing 1

This analysis is similar to the previous one where in this case, the same derived hypothesis was used. The only difference in this analysis is the data samples were derived using the collected two parameters of expected work and the actual work. In this analysis, the successful achievement levels of the scheduled workloads were used to evaluate the two methods. Once again, the same hypothesis with the LOC analysis was used as follows.

Null hypothesis (H_0) – Agile process developer productivity cannot be improved through applying Lean principle

Alternative hypothesis (H_1) – Agile process developer productivity can be improved by using Lean practice techniques

For a given week (W_i) and a given group (G_i), the actual work level was calculated considering the work hour measures. For this analysis, the work done on defect fixing time was not considered, since that time was not scheduled explicitly, in the expected work norms. However, with the results it was obvious that there had been a direct impact from the defect fixing work to the actual work level, making it further deviate from the expected work norm.

Since the analysis was based on achievement level of the scheduled work for a given week, the following model (3) was used to derive the required sample information for the hypothesis testing.

$$\frac{(\text{Actual work level - Expected work level})}{\text{Expected work level}} * 100\% \tag{3}$$

Having a positive value as the output from this equation means, in that particular week, the actual work done has exceeded the scheduled or expected work amount. A negative value indicates that the actual work done is less than that of the expected. The values were considered as percentages for the comparison and analysis convenience.

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
G1	-39%	-27.5%	-17%	-28%	-9.6%	-7%	-34%	-18.7%	-24.7%	-22.3%
G3	-30%	-23.3%	-31.4%	-27.7%	-16%	-16%	-38%	-28%	-16%	-13%
G4	-29%	-25.8%	-34%	-32%	-23.1%	-17.3%	-16.2%	-12%	-15%	-17.6%
G9	-17.3%	-18%	-35.6%	-22.7%	-18%	-29.3%	-16%	-4%	-15.1%	-27.2%
G10	-14.7%	-16%	-33.3%	-9.3%	-21%	-29.2%	-12%	-1.2%	-23.2%	-17.7%

Table 5. Sample A – deviation percentage from expected work amount

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
G2	-42%	-34%	8%	-24%	-21.6%	-8%	-15.2%	-19%	-2.7%	-40%
G5	-20%	-20%	-2.7%	1%	-18%	-1.5%	-14%	-24%	-16	-13.8%
G6	-14.4%	-2%	-12%	-9%	-3.8%	-5.7%	-26%	-58.7%	-5.3%	-11.7%
G7	-10.6%	-28%	-9.3%	-18%	-5%	-13.3%	-17.3%	-5.3%	-30%	-22.2%
G8	-16%	-16%	-7%	-12%	-15.3%	-20%	-26.7%	-17.3%	-32%	-13.3%

Table 6. Sample B – deviation percentage from expected work amount

The above two tables (Table 5 and 6) show the deviation percentages from the expected work amount for the two samples. These data samples used to test the hypothesis using ANOVA method as done in the previous analysis.

```
Welcome to Minitab, press F1 for help.
Retrieving project from file: C:\DOCUME~1\A\DESKTOP\MINITABACT.MPJ
```

One-way ANOVA: C1, C2

```
Analysis of Variance
Source      DF        SS        MS         F         P
Factor       1    0.0672    0.0672      6.07     0.016
Error       98    1.0864    0.0111
Total       99    1.1536
                                    Individual 95% CIs For Mean
                                    Based on Pooled StDev
Level        N      Mean     StDev  --+---------+---------+---------+----
C1          50   -0.2140    0.0882   (---------*---------)
C2          50   -0.1621    0.1199                     (---------*---------)
                                    --+---------+---------+---------+----
Pooled StDev =   0.1053           -0.240    -0.210    -0.180    -0.150
```

Fig. 6. MINITAB output - ANOVA output for hypothesis test on work levels

According to the obtained results from the ANOVA test, following information was obtained for the analysis.

For the sample A: The mean value μ_A = -21.4% and the standard deviation σ_A = 8.82.
For the sample B: The mean value μ_B = -16.21% and the standard deviation σ_B = 11.99.

According to the used equation model, having a higher (towards to the positive values) mean value is better since the deviation from the expected work level is lesser. Furthermore, statistically the ANOVA test resulted in the p-value as 0.016; i.e. *p< 0.05*. This means that the Null hypothesis (H_0) can be rejected with 95% confidence level, based on achieving the expected effective work level for a given time period. For the work hour measures, since the additional tasks were not incorporated, clearly, the blended process practice allows higher productivity with a higher effective work level. This implies that the Agile process developer productivity can be improved by incorporating Lean practices along with developers' usual Agile process tasks.

6.3 Analysis – Defect Rate Behaviour
Apart from the hypothesis testing, the defect fixing rate was also analyzed for the two samples through the examined time period. A defect was classified as an unexpected or erroneous behaviour of a selected piece of module or component, which has been already compiled successfully and committed to the project. With that respect, compilation errors of the code were not considered as defects. The main intention for this analysis was to examine the supportability level towards the work perfection of the two paradigms.
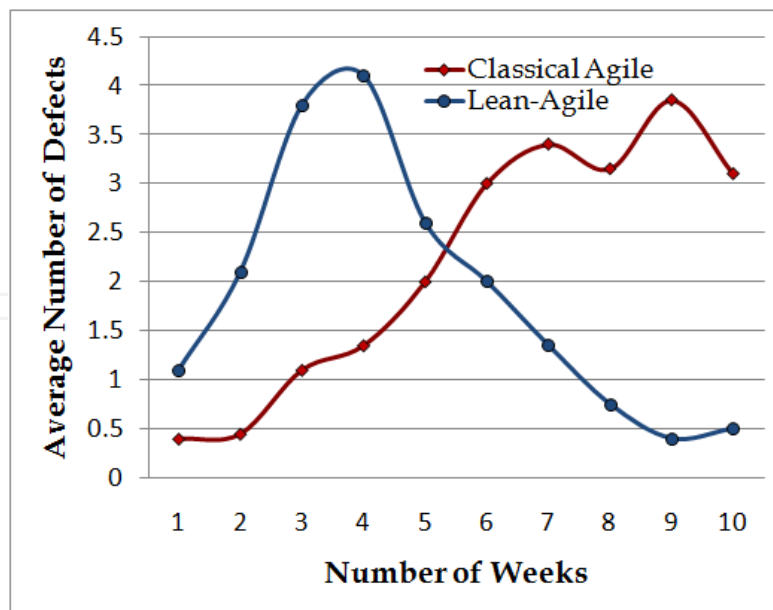
Fig. 7. Average defects rates in the examined time period for the two samples

A significant pattern difference between the defect rates of the two samples was observed during the experiment period, as shown in Figure 7. A higher rate for Lean-Agile sample at the early stages of development was due to that their autonomous and value perfection norms with the development. On the other hand, the classical Agile groups did not find many defects during the early weeks, since they did not pay much attention to the perfection of what they were developing. At the later stages, this situation swapped between the two samples and the Lean-Agile practice seems to have a stable minimal defect rate, where as the classical Agile practice has experienced a high and varying defect rate. A possible explanation to this behaviour is that the unfixed hidden defects in components from early developments would cause emerging defects once they integrated each other. Importantly, having lesser defects in the later stages is very essential for the stability of the project and to be aligned with the project schedule. Furthermore, defects emerge in later stages are relatively expensive to fix than that of the early stages. However, the average defect rates per week for the two samples were close to each other as (Sample A) μ_A = 2.2, and (Sample B) μ_B = 1.92. If the study was extended further, this closeness would have changed since the group A is getting further defects with its trend. However, based on the available information it can be concluded that applying Lean principles stabilizes the Agile development phase with respect to quality and perfection, especially in later stages of the development phase.

6.4 Experiment Limitations
There have been few experimental limitations were identified with this research, which are mentioned below. However, their impact to the result was not significant enough to create externalities among the data samples; hence to the outcome. One of the key limitations observed with the experiment was that the assumed identical skill level among the students in the sample. It is a known fact that no two humans can have identical skill levels. However, this fact is a generic limitation to all the experiments, which involve human skill based activities. The best possible scenario one could achieve is to have nearly similar

skilled people within the sample, i.e. minimize the skill differences as much as possible. In that regard, the selected experiment population is one of the best cases one could find for such an experiment. The reasons for such a strong statement were discussed under the experiment rationalization section. Therefore, the impact of this limitation was minimal to the study.

Another limitation with the study was the truncation errors of the collected data. Literally, what have happened to be the developers were confident on expressing their values with integer figures of hours without the decimal or fractional values. For an example, they might have said their actual work amount as 23 hours, but the precise value may be 23.2 hours or 22.7 hours, etc. This was with the LOC measures as well. If there were extreme cases, which questioned the accuracy of the data additional parameters such as compile time and codebase log files, were used to cross validate the claimed figures, as a sanity check. However, since this is common to both samples of the experiment this was nullified at the end. Furthermore, this type of truncation errors have the normal distribution behaviour where the standard error mean is 0; i.e. the impact at the population level is insignificant.

Another limitation was the domain differences between the projects. Sometimes, domain specific knowledge can be a significant factor for project success. Some of the projects were in different domains, which introduced some impact to the experiments. However, since students have already followed their literature survey and background studies, at the time they engage with software development, every group had a sufficient level of competence on their respective domains, resulted in lesser impact to the experiment outcome.

7. Conclusion

This research has introduced significant policy implications to Agile practitioners. First of all, software development activities which follow Agile process, can be considerably benefited through using the proposed process model. In fact, the proposed process model successfully, creates more value oriented, certain, value streamed, and productive software development environment over the classical Agile approach. The research results also reveal a more defect free development activity, essentially in the crucial stages of the development. Importantly, the proposed blended process shows more stability over frequent requirement changes, which is inevitable within an Agile process based software development. The used Lean principles have acted as stabilizing agents within certain Agile practices.

Another possible implication derives from this study is that, like the proposed process practice improves the development works within the software development phase, there is a significant potential to improve the other software lifecycle phases, such as, Requirement Engineering, Design, Testing, and Deployment, even though they are less visible within the Agile practices. In fact, more dominancy on development phase alone, has made the Agile practices more vulnerable to process instability, frequent changes and overhead development works. With the Lean practices, Agile process can have short yet steady Requirement Engineering, Design and Testing phases without affecting to the main development works.

Moreover, the recent hype on Agile manufacturing can also be benefited from the amalgamation of suitable Lean concepts as required. This means, though this study was mainly focused on software industry, it is possible to extend the proposed process model as required for other industries of interest. Specially, the industries of promising future with

Agile manufacturing, could be enhanced the process potentials resulting in fruitful returns. Moreover, the flexibility given in the proposed process model allows practitioners to customize their practices as per the industry norms without reducing the benefits.

It is required a further examine on this proposed process model in a broad spectrum of industrial environments and formulate a standardized process practice for the proposed model. It is crucial to substantially practice the model in a wider range of projects in diversified environments to fine tune the proposed practices. Therefore, it is expected, thus encourage industrial practitioners to use this model widely while interested researchers to research further to improve, standardize and make popular for the benefit of Agile practitioners.

8. References

Abrahamsson, P., Babar, M. A., Kruchten, P., (2010), Agility and Architecture: Can They Coexist?, *IEEE Software*, Vol. 27, No. 2, March/April 2010, pp. 16-22, IEEE Press

Agile Manifesto, (2001), Manifesto for Agile software development, [available at] http://Agilemanifesto.org/, [accessed on 19th December 2009]

Augustine, S., (2005), *Managing Agile Projects*, Robert C. Martin series, Prentice Hall Publishers

Basili, V., (1993) , The Experimental Paradigm in Software Engineering," *in LNCS 706, Experimental Software Engineering Issues: Critical Assessment and Future Directives*, H.D. Rombach, V. Basili, and R. Selby, eds., *Proceedings of Dagstuhl-Workshop, September 1992*, Springer-Verlag,.

Basili, V., (2007), The Role of Controlled Experiments in Software Engineering Research, *in Empirical Software Engineering Issues, LNCS* 4336, V. Basili et al., (Eds.), Springer-Verlag, pp. 33-37

Black, S., Boca, P.P., Bowen, J.P., Gorman, J., Hinchey, M., (2009), Formal Versus Agile: Survival of the Fittest?, *Computer, IEEE Press*, Vol. 42, pp. 37-45

Cockburn, A., Highsmith, J., (2001), Agile software development: the people factor, *IEEE Computer*, pp 131-133.

Chow, T., Cao, D., (2008), A survey study of critical success factors in Agile software projects, *Journal of Systems and Software*, Vol. 81, Issue 6, pp. 961-971

Cohen, D., Lindvall, M., Costa, P. (2003), *A State of the Art Report: Agile Software Development*, Data and Analysis Center for Software 775 Daedalian Dr. Rome, New York 13441-4909, p. 01

Danovaro, E., Janes, A., Succi, G. (2008), Jidoka in software development, *In Companion To the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications, OOPSLA Companion '08. ACM*, pp. 827-830.

Deek, F. P., McHugh J. A. M., O. M. Eljabiri, (2005), *Strategic Software Engineering an Interdisciplinary Approach*, Auerbach Publications, FL, USA, p. 94

Fatina, R., (2005), *Practical Software Process Improvement*, Artech House, Boston, p. 06

Fuggetta, A., (2000), Software Process: A Roadmap, *in Proc. of the Conference on the Future of Software Engineering, ICSE*, Limerick, pp. 25-34

Gross, J. M., McInnis, K. R., *Kanban Made Simple: Demystifying and Applying Toyota's Legendary Manufacturing Process*, AMACOM, 2003

Hibbs, C., Jewett, S., Sullivan, M., (2009), *The Art of Lean Software Development: A Practical and Incremental Approach*, O'reilly Media, CA, USA

Humphrey, W. S., (2006), *Managing the Software Process*, SEI, Pearson Education, India, p. 03

Jacobs D., (2006), *Accelerating Process Improvement Using Agile Techniques*, Auerbach Publications, FL, USA

Kupanhy, L., (1995), Classification of JIT techniques and their implications, *Industrial Engineering*, Vol. 27, No.2

Lee, G., Xia, W., (2010), Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data, *MIS Quarterly*, Vol.34, No.1, pp.87-114

Middleton, P., (2001), Lean Software Development: Two Case Studies. *Software Quality Journal*, Vol.9, No.4, pp. 241-252

Middleton, P., Taylor, P. S., Flaxel, A., Cookson, A., (2007), Lean principles and techniques for improving the quality and productivity of software development projects: a case study, *International Journal of Productivity and Quality Management*, Vol. 2, No. 4, Inderscience publishers, pp. 387-403

Miller, L. Sy, D. 2009. Agile user experience SIG, *In Proc. of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems, CHI '09.* ACM, New York, NY, pp. 2751-2754

Narasimhan, R., Swink, M., Kim, S.W., (2006), Disentangling leanness and agility: An empirical investigation, *Journal of Operations Management*, Vol. 24, No.5, pp. 440–457

Naylor, J.B., Naim, M.M., Berry, D., (1999), Leagility: Integrating the Lean and Agile manufacturing paradigms in the total supply chain, *International Journal of Production Economics,* Vol. 62, No. (1/2), pp. 107–118.

Ohno, T. (1988), *Toyota Production System: Beyond Large-Scale Production*, Productivity Press, Cambridge, MA, USA

Oppenheim, B. W., (2004), Lean product development flow, *Systems Engineering*, Vol.7, No. 4, pp. 352-376

Perera, G.I.U.S., (2009), Impact of using Agile practice for student software projects in computer science education, *International Journal of Education and Development using Information and Communication Technology (IJEDICT)*, Vol. 5, Issue 3, pp.83-98

Perera, G.I.U.S. and Fernando, M.S.D. (2007), Bridging the gap – Business and information systems: A Roadmap, *In Proc. of 4th ICBM conference*, pp. 334-343.

Perera, G.I.U.S. and Fernando, M.S.D. (2007), Enhanced Agile Software Development — Hybrid Paradigm with LEAN Practice, *In Proc. of 2nd International Conference on Industrial and Information Systems, ICIIS 2007*, IEEE, pp. 239 – 244.

Perera, G.I.U.S. & Fernando, M.S.D., (2009) Rapid Decision Making For Post Architectural Changes In Agile Development – A Guide To Reduce Uncertainty, *International Journal of Information Technology and Knowledge Management*, Vol. 2, No. 2, pp. 249-256

Petrillo, E. W., (2007), Lean thinking for drug discovery - better productivity for pharma. *DDW Drug Discovery World*, Vol. 8, No.2, pp. 9–16

Poppendieck, M., (2007), Lean Software Development, *29th International Conference on Software Engineering (ICSE'07)*, IEEE Press

Poppendieck, M., Poppendieck, T., (2003), *Lean Software Development: An Agile Toolkit* (The Agile Software Development Series), Addison-Wesley Professional

Prince, J., Kay J.M., (2003), Combining Lean and Agile characteristics: Creation of virtual groups by enhanced production flow analysis, *International Journal of Production Economics*, Vol. 85, No. 3, pp. 305–318

Rozum, J. A., (1991), Defining and understanding software measurement data, *Software Engineering Institute*,

Salo, O., Abrahamsson, P., (2005), Integrating Agile Software Development and Software Process Improvement: a Longitudinal Case Study, *2005 International Symposium on Empirical Software Engineering, IEEE press,* pp. 193-202

Santana, C., Gusmão, C., Soares, L., Pinheiro, C., Maciel, T., Vasconcelos, A., and A. Rouiller, (2009), Agile Software Development and CMMI: What We Do Not Know about Dancing with Elephants, P. Abrahamsson, M. Marchesi, and F. Maurer (Eds.): *XP 2009, LNBIP 31*, Springer-Verlag, Berlin Heidelberg, pp. 124 – 129

Shalloway, A., Beaver, G., Trott, J. R., (2009), *Lean-Agile Software Development: Achieving Enterprise Agility*. 1st. Addison-Wesley Professional

Sugimori, Y., Kusunoki, K., Cho, F., Uchikawa, S., (1977), Toyota production system and Kanban system: materialisation of just-in-time and respect-for-human system, *International Journal of Production Research,* Vol. 15, No.6, pp.553–564.

Syed-Abdullah, S., Holcombe, M., Gheorge, M., (2007), The impact of an Agile methodology on the well being of development teams, *Empirical Software Engineering*, 11, pp. 145–169

Udo, M., Vaquero, T. S., Silva, J. R., and Tonidandel, F., (2008) Lean software development domain, *In Proc. of ICAPS 2008 Scheduling and Planning Application workshop*, Sydney, Australia

Vokey, J. R., Allen S. W., (2002), *Thinking with Data*, 3rd Ed., PsyPro, Alberta

Womack J. P., Jones, D.T., (2003), *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, New Ed., Free Press, UK

Yusuf, Y.Y., Adeleye, E.O., (2002), A comparative study of Lean and Agile manufacturing with a related survey of current practices in the UK, *International Journal of Production Research*, Vol. 40, No.17, pp. 4545–4562.

Future Manufacturing Systems

Edited by Tauseef Aized

This book is a collection of articles aimed at finding new ways of manufacturing systems developments. The articles included in this volume comprise of current and new directions of manufacturing systems which I believe can lead to the development of more comprehensive and efficient future manufacturing systems. People from diverse background like academia, industry, research and others can take advantage of this volume and can shape future directions of manufacturing systems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following: