HARVARD | BUSINESS | SCHOOL

Lean Principles, Learning, and Software Production: Evidence from Indian Software Services

Bradley R. Staats
David M. Upton

Working Paper

08-001

Lean Principles, Learning, and Software Production: Evidence from Indian Software Services

Bradley R. Staats*
Harvard Business School
Morgan 428B
Boston, MA 02163
Tel: 617 496-1462
Fax: 617 496-4066
bstaats@hbs.edu

David M. Upton
Harvard Business School
Morgan 415
Boston, MA 02163
Tel: 617 496-6636
Fax: 617 496-4066
dupton@hbs.edu

Confidential Draft:
*** Corresponding Author**

Acknowledgments

We would like to express our gratitude to Alexis Samuel, Sambuddha Deb, Ravishankar Kuni, Seema Walunjkar and many other individuals at Wipro for their significant commitment of time and effort for this project. This paper benefited substantially from the thoughts and comments of Kent Bowen, David Brunner, Vishal Gaur, Robert Hayes, Ananth Raman, Zeynep Ton, and Noel Watson. We are grateful to the Division of Research of Harvard Business School for generous funding. All errors remain our own.

Lean Principles, Learning, and Software Production: Evidence from Indian Software Services

Abstract While the concepts of lean production are frequently applied in service organizations there is little work that rigorously has examined implementing lean production in contexts other than manufacturing as well as lean production's impact on performance in these settings. In this paper we set out to accomplish both tasks by investigating the implementation of a lean production system at an Indian software services firm. Combining a detailed case study and empirical analysis we document the internal processes that the lean initiative influences. We find that lean projects perform better than the non-lean projects in our sample in many, but not all cases. Building on this result we see that the impact of the techniques on problem solving, coordination, and standardization of work improve the way that the firm learns as well as its productivity. In so doing, we gain insight into how a company can build an operations-based advantage.

Keywords: Lean production, operations strategy, organizational learning, software

1. Introduction

Lean principles, or the tenets of the Toyota Production System (TPS), continue to be of great interest to the operations community. Many credit Toyota's sustained success to the tireless application of these ideas to their manufacturing and management systems. This has provided an incentive for many manufacturing companies to imitate, either wholesale or in part, "lean" principles in their improvement programs. The application of lean production in the services is a more recent manifestation of the use of these principles. However, the utility and impact of such ideas in non-manufacturing contexts remains a contentious issue, leaving many managers to wonder if they are merely applying inappropriate, faddish, ideas with others arguing that lean principles have universal applicability (C.f. Sousa and Voss 2001). In manufacturing itself, lean production has led to improved performance in many cases (Li et al. 2005; Shah and Ward 2007) yet failed implementations are not uncommon, and as Shah and Ward (2007) note, the general understanding of lean production is mixed.

In this article, we examine the application of lean production to software services. We report our observations and analysis of a particular implementation at a large Indian software vendor. We first describe the theoretical basis for our study, and then present empirical results in both descriptive and quantitative forms. In so doing, our objective is both to examine rigorously the implementation of lean production in services and to explore lean production's impact on performance in this context.

.

In 2004 Wipro Technologies, an Indian outsourced software services provider faced a quandary. Senior managers believed that the marginal benefits of its previous quality initiative were exhausted and they wanted a new approach to improve quality while building innovative capabilities. However, there were no new, pre-existing improvement methodologies to implement within the software realm. After searching across industries, they decided that the ideas that had served Toyota so well might also create an opportunity for improvement in their own business.

In mid-2004, the company quietly launched a pilot "lean" initiative: an endeavor that attempted to translate ideas on lean production from manufacturing to software services. A core team was formed and spent several months reading, visiting companies practicing lean manufacturing, and discussing ideas for implementation. The team then began the execution of the initiative. While continuing their regular jobs, each team member sought out a client-facing software project in which to implement a lean approach to software services. Eight of the ten projects selected were successful (>10% improvement on the pre-specified metric). Buoyed by the success of the pilot, Wipro rolled out the program across the firm. By June 2007 Wipro had 772 lean projects completed or underway. Our analysis shows that lean projects performed better, and with lower variation than a matched comparison set in many, but not all cases.

Prior theorists note that implementations of lean production may vary greatly across different manufacturing settings due to contextual differences (de Treville and Antonakis 2006). By extension then, moving the ideas of lean production to services requires a process of exploration. Despite the many calls by practitioners for lean services, there is a lack of work examining theoretically why lean production may be valuable in services and verifying this question empirically. To accomplish this task we exploit the unique opportunity that Wipro offered. Since few preconceived notions about lean production exist in the software setting, it provides the opportunity to examine core ideas that are essential for success.

Traditional manufacturing-based lean artifacts (Shah and Ward 2003; 2007) were not directly applicable, therefore Wipro had to generate hypotheses as to the role of these artifacts and then test them in practice. For example, the idea of an andon cord is not easily replicable in software as there is no line

to stop. However, the principle of generating a pre-specified call for help when there is a deviation, thereby keeping the problem and solution together in time, space and person has considerable power (Spear and Bowen 1999). This informed trial and error approach increases our understanding of the mechanisms by which lean production improves performance in services.

In the following section we examine the literature on the principles of lean production. In section 3 we introduce our setting and methodology before exploring the relevance of lean production to software services in section 4. Next, in section 5, we describe the practices implemented at Wipro in order to understand how the approach changes the processes through which the company delivers customer value and how this impacts learning and performance. In section 6 we compare the performance of lean and non-lean projects empirically. Since lean management has many traits of a management fad, documenting improved performance is important (Abrahamson and Fairchild 1999). Finally, we examine the challenges of translating lean production to software services, and other service-based settings.

2. Principles of Lean Production

The Toyota Production System (TPS) did not emerge fully formed from the vast ocean of improvement methods to lead Toyota to international success. Rather it was built through a series of small changes over many years (Fujimoto 1999). Early on, Toyota leaders did not have and believed that they could not attain the economies of scale of Ford or General Motors and so they tried to develop a system which Henry Ford might have used in their situation (Ohno 1988).

In the early 1980s, as Toyota and other Japanese manufacturers made inroads in global markets, the call was sounded to study these companies in depth (e.g. Hayes 1981). This led to books on TPS by its creators as well as the launch of programs to study lean principles at multiple universities (e.g. Monden 1983; Ohno 1988; Womack et al. 1990; Clark and Fujimoto 1991). In an attempt to generalize the work of Toyota for other manufacturing settings, Krafcik (1988) coined the term lean. Lean was chosen to highlight the principles of limiting inventory, excess workers, or "waste", as opposed to other auto manufacturers' "buffered" approaches (Hopp and Spearman 2004).

Despite significant study, as Shah and Ward (2007) note, the field has struggled with a lack of clarity about what is and what is not lean production. There has been significant recent work defining the "how" of lean production in manufacturing (MacDuffie 1995; Shah and Ward 2003; Narasimhan et al. 2006; Shah and Ward 2007). As noted above, since bundles of practices have not been identified in services, as they have in manufacturing (Shah and Ward 2003; 2007), we seek to examine the Wipro implementation through the lens of the work that has identified lean principles.

Two early works on lean principles were Taiichi Ohno's (1988) *Toyota Production System: Beyond Large-Scale Production* and Womack and Jones' (1996) *Lean Thinking*. Taiichi Ohno, one of the creators of TPS, states that there are two criteria for lean production: just in time (JIT) and autonomation. JIT is a pull system where production at each step does not begin until it is signaled for by the customer (the downstream step). To support JIT, Ohno developed the concept of a kanban, with six accompanying rules. Autonomation refers to autonomous automation. The goal is not to eliminate humans from production, but rather to focus them on the highest value parts of the practice.

Lean Thinking, arose out of the work of the MIT International Motor Vehicle Program (IMVP). While studying all aspects of the auto industry, the IMVP focused specifically, on the production of automobiles. In addition to the prolific output of Womack and Jones (1990; 1994; 1996; 2005), others associated with the IMVP include Krafcik (1988), MacDuffie (1995; 1996; 1997) and Cusumano (1998).

In *Lean Thinking,* Womack and Jones (1996) narrow the concepts of lean management into five categories: value, value stream, flow, pull and perfection. Value defines the use the product provides to a customer and works backwards to build the production process. Firms map production (create a value stream) to ensure that each step provides value. Flow reorganizes processes so that products move smoothly through the value creating steps. Pull involves each customer calling output from the previous step on demand (see Hopp and Spearman 2004 for a critique of this principle, p. 141). Finally, perfection requires constant striving to meet the customer's needs and improve one's process – with zero defects.

As noted by Hopp and Spearman, Ohno's book was "clear on basic philosophy", but "general" while Womack and Jones provided examples, but "did not provide clean definitions of basic concepts," perhaps even distorting Ohno's intentions in their principles (Hopp and Spearman 2004: 140-141).

Spear and Bowen (1999) stepped into this gap with the explicit goal of identifying the principles of TPS inductively. Spear conducted an ethnographic study in which he visited 33 sites of Toyota and its suppliers (Spear 1999). From this effort and analyses four rules for TPS were derived:

> ***Rule 1:*** All work shall be highly specified as to content, sequence, timing, and outcome.
> ***Rule 2:*** Every customer-supplier connection must be direct, and there must be an unambiguous yes-or-no way to send requests and receive responses.
> ***Rule 3:*** The pathway for every product and service must be simple and direct.
> ***Rule 4:*** Any improvement must be made in accordance with the scientific method, under the guidance of a teacher, at the lowest possible level in the organization (ibid: 98).

The rules capture four essential points about lean systems: the need for standardization of work, the utility of direct communication, the role of direct architecture, and the approach to problem solving. It is the combination of these factors into a system, that enables Toyota's performance and makes TPS so hard to replicate. Fujio Cho, Chairman of Toyota says, "what makes Toyota stand out is not any of the individual elements… [but] having all the elements together as a system." (Liker 2004: xv)

This brief review does not fully survey the literature on lean production, but rather focuses on the work that has been done to identify its underlying principles (See Hopp and Spearman 2004; Shah and Ward 2007 for thorough surveys)[1]. From Spear and Bowen (1999) we learn that the difference in a lean production system comes from how it alters the way a company learns through changes in its approach to problem solving, coordination through connections and pathways, and standardization. In the next section we describe the setting and the methodology used in our study.

3. Setting & Methodology

3.1. Setting

Wipro Technologies is an outsourced software services provider. The company's global service offerings include application development, engineering services, IT infrastructure management, testing,

[1] Related work that originated around product development practices, but also considers production is not mentioned in these works, but is germane (e.g. Clark and Fujimoto 1991; Ward et al. 1995; Fujimoto 1999; Liker 2004).

and maintenance. Wipro is diversified across technologies, software languages, and industries. In June 2007 the company had over 72,000 employees and annualized revenues greater than $4 billion.

The software services product delivery process can be broken down into three steps: advise, design, and execute as shown in Figure 1.

**********INSERT FIGURE 1 ABOUT HERE**********

Indian firms started their businesses with the execution of non mission critical projects, such as maintaining existing systems. Early work involved sending employees to the customer's site to perform work at the customer's direction. Steadily firms built processes to perform more work offshore (in India). Next they expanded into design. Design consists of taking a conceptual idea generated in the advise stage and building a model of a system that includes its components and their relationship to each other at an abstraction level that serves as the input for execution (Ulrich 1995). Finally, they began moving into advising – assessing a customer's stated and latent needs to guide their information technology strategy.

In 2004 Wipro management was faced with both productivity and innovation challenges. The cost of the company's key internal resource, software engineers, was increasing by 10+% per year while the company was undergoing high growth in the number of employees (30+% annually) with high employee turnover (10-20% annually for the industry) (NASSCOM 2005). This rapid intake and turnover of employees meant that the company needed robust processes to provide support and monitoring for engineers and managers. Also, demand factors were impacting the business as customers were exerting price pressure while requiring improving quality (McCarthy 2004).

In the past, in part due to customer pressures for certification, Wipro relied on public process improvement measures such as ISO 9000 and the Capability Maturity Model (CMMI). Yet, the same strengths that helped Wipro replicate their systems as they grew were imitable by competitors (Winter and Szulanski 2001). Wipro wanted to find a new approach that would also provide an operations-based advantage and a barrier to imitation (Hayes and Pisano 1996).

In addition to a need for increased productivity, changing customer demands were putting pressure on the company's innovative capabilities. Supplier quality (in terms of defects / lines of code)

was no longer a competitive differentiator, but a qualifier to be in the business (Hill 1993). Customers were asking Wipro to deliver quality code *and* a business advantage. One senior quality leader remarked:

> Our customers tell us that, 'you are very good at doing what I tell you to do. If I ask you to build a three-legged table then you just do it, but you never ask why the table has only three legs.' Our customers want us to deliver innovative ideas … [and] they know when Wipro is not delivering.

Another manager added, "We need to link business focus and process focus. Through things like CMMI, we have gotten very good at managing processes. However, we don't really know how to manage people." (C.f. Adler 2005)

With these considerations in mind, in early 2004, management at Wipro decided to identify and implement a new process improvement approach to improve deliverable quality while building innovative capabilities. An internal process identified three candidates: lean production, the Malcolm Baldridge Quality Award, and TRIZ[2] (Altshuller et al. 1999). They decided that Baldridge was too focused on quality and TRIZ was too focused on innovation, but the lean principles offered the right balance. While Wipro personnel were not the first to think of applying lean principles to software development they found the outside thinking unsuitable as it was not at an industrial scale (Poppendieck and Poppendieck. 2003; Middleton and Sutton 2005).

The five person Productivity Office (PO) within Wipro was tasked with leading the introduction of the lean initiative. They were responsible for learning about lean production in manufacturing, training, monitoring progress, and sharing best practices. A member of the PO was assigned to support all lean projects and conduct monthly review meetings. Lean projects were closed if their associated teams did not make sufficient effort to implement lean principles. Typically lean projects were closed because the project's objectives were changed substantially (e.g. ideas from lean production were going to be applied to improving testing, but the customer eliminated the testing portion of the project). A lean project could not be closed because the project was progressing poorly. For a project to be defined as a lean project, a team received training and then was required to make a good faith effort to apply the ideas on lean production from this training, including at least one countermeasure.

[2] Теория Решения Изобретательских Задач (ТРИЗ) – Theory of Inventive Problem Solving

3.2. Methodology

This study followed established case study methodologies for data collection and analysis (Eisenhardt 1989; Yin 2003). The work comprised two stages. In the first, from January 2005 to January 2006, we visited Wipro's development centers in Bangalore, India three times, and spent five weeks onsite over these visits. The fieldwork began four months after the lean effort was launched so we could discuss events as they unfolded with participants. Interviews followed a semi-structured format designed to elicit consistent information across respondents while remaining flexible to explore each interviewee's unique perspective. To examine the impact of the lean initiative on the entire organization we spoke with people at all hierarchical levels within Wipro. Therefore, we conducted interviews with senior management, business unit leaders, account and project managers, and team members.

Interviews lasted from one to two hours and some key respondents were interviewed multiple times. Interviews were recorded when possible and extensive notes were taken during all interviews. In total, sixty-six individuals were interviewed. In addition to interviews, we attended lean project review sessions, a lean training session, and a quarterly review session with Wipro's internal quality group. During and after interviews we collected supplementary materials such as written project updates and lean concept notes written by the Productivity Office. After each visit we categorized the data based on the four principles identified in the literature review. A priori there was not a target marking the end of stage one, but the process was continued until reaching category saturation – the point at which new evidence did not appear (Strauss and Corbin 1990). This occurred during the third visit.

Stage two of the study took place from February 2006 to July 2007 during which we made three further visits to Wipro's development centers. During this stage we returned to earlier interview subjects to discuss how the lean initiative had progressed. In addition to the qualitative analysis, we collected and analyzed quantitative project data for both lean and non-lean projects. In the next section we discuss how the principles of a lean system impact learning and performance within software services.

4. Lean Principles and Software Services

Spear and Bowen (1999) propose that lean production alters learning through changes in problem solving, coordination through connections and pathways, and standardization. We now examine each of these concepts in more detail and discuss their relevance to software services.

4.1. Problem Solving

Problem solving provides the foundation for what a firm does (Nelson and Winter 1982). In TPS problem solving is done at the lowest level possible and involves hypothesis testing to move the organization towards the ideal. The first point ensures that individuals who are responsible for work are involved in its improvement and imposes a teaching responsibility on managers. The latter point highlights that organizations should structure problem solving like scientific experiments in which an overall objective lays out the course for change while specific hypotheses drive the individual changes (Spear and Bowen 1999; Spear 1999).

To explore how problem solving impacts software services, we use MacDuffie's (1997) model for problem solving: (1) problem definition, (2) problem analysis, (3) generation and selection of solutions, (4) testing and evaluation of solutions, (5) routinization.

Different approaches to problem definition change what is considered a problem and how a solution is sought (MacDuffie 1997). The question of how one knows when there is a problem in software is not trivial. At the end of the process, if the software does not work then it is straightforward to identify a problem. This is an expensive and inefficient way to troubleshoot (Boehm 1981). Therefore, the goal for problem definition in software is to find problems earlier.

We look at steps two through four together since they may occur jointly in development. There are at least four relevant categories of problems in software services: customer, project-specific, team, and individual. Addressing each category can speed problem solving and generate learning within and across projects. First, customer uncertainty, the challenge of identifying a customer's objectives, is often more severe in software services than technical uncertainty (MacCormack and Verganti 2003). One manager noted, "A lot of customers aren't clear what they want, it is often evolving." Techniques to draw out this

customer information may prove valuable. Project specific learning may occur from prioritization or reordering of work. Team learning may arise from practices that highlight unpredicted interdependence. Finally, problems are often not known until long after code is written. A process that keeps problems and solutions together in time and space may increase individual problem solving and learning.

The fifth step in problem solving, routinization may also improve problem-solving and learning. By establishing processes for problem solving, an organization can trigger cycles of learning which may lead to continuous improvement and help them to avoid inertia (Zollo and Winter 2002). The important point is not necessarily getting the right answer the first time, but rather doing it in the right way (Clark and Fujimoto 1991). Tucker (2004) shows that people often do not learn from errors because they execute work-arounds to complete their task – by focusing on immediate problem solving they forgo the chance to learn. By routinizing proper processes, a group may counteract this effect.

4.2. Coordination – Connections and Pathways

Coordination in a lean system is direct and simple, and can be broken into two constituent parts: connections and pathways. A connection describes the linkage of two individuals in the system, while a pathway consists of the connections, which make up the flow of goods, services, or information through the organization. There are three themes for coordination in TPS (Spear and Bowen 1999; Spear 1999).

First, TPS highlights the importance of architectural simplification. In addition to the benefits of reduced complexity, with direct connections and simple pathways, the product/service flows of the organization are integrated with the information flows of the organization. Keeping the two together assists problem solving and makes change easier and more reliable as both can be altered simultaneously. The process which goods/services follow is streamlined so team members can verify the necessity of each step in the process and minimize total linkages so information flows smoothly from customer to supplier.

The second theme is that of information clarity. Communications are streamlined so there are neither too few nor too many signals. With direct, clear connections and pathways, individuals spend less time thinking about what to do and instead focus on how to accomplish the task. For example, Toyota's

workers are directly connected with their team leaders through the andon cord, so a pull on the cord by a worker signals to one person, his team leader, that a hypothesis has failed and that he needs help.

The final theme is problem identification. Each connection or pathway should be self-diagnostic – i.e. hypotheses are checked and revised if false. Returning to the andon cord example, we see that the cord permits a clear signal to pass between the worker and team leader, and it assists in identifying that a problem has occurred. By specifying connections workers identify their customer and suppliers and then can appreciate more clearly the capabilities and opportunities for improvement of each (Spear 1999).

Coordination plays a central role in software services. It is rare that work can be fully partitioned to eliminate interdependencies (von Hippel 1990; Kraut and Streeter 1995). Therefore, practices which identify and streamline connections and pathways, both in terms of people and within the code itself, offer important benefits. Focus on coordination is particularly important in outsourced software services since often the team is distributed between India and a customer's site (typically North America or Europe).

4.3. Standardization

Tasks should be standardized for two primary reasons. First, standardization permits continuous hypothesis testing. Once the work for a task is specified as to substance, order, timing, and result, then two hypotheses are tested every time the work is completed: (1) whether the individual completing the task is capable of doing so; (2) whether the activity will produce a quality output. If either hypothesis is rejected, then problem solving cycles are triggered (Spear and Bowen 1999; Spear 1999).

Second, standardization makes it easier to identify cause and effect. When work is standardized and its actual condition is compared to its expected condition immediately after the work is completed then the opportunity for improvement increases. Since knowledge about a task often comes from the completion of the task then the collocation of the action and the information enables more effective learning and improvement (Spear 1999). Additionally, standardization reduces variance so the problem solver is more likely to notice that there has been a problem and to identify the underlying causal relationship, which assists in the improvement effort (Clark 1988; Bohn 1995). For standardization to be valuable then it cannot be a static process, rather standards must be frequently updated (Hino 2006).

The delivery of software services is a complex process which varies by project depending on factors such as project size and the customer's context. Software development typically addresses an underconstrained problem in which equifinality is an issue. There are many paths to develop functional software and it is often difficult to judge the final output with more than rough measures (such as 'does the product work? was it delivered on time? was it delivered on budget?'). In this setting there are many benefits to process standardization which defines the necessary work and derives constant means for accomplishing it. Given the variability in software services a desired process must be neither too rigid nor too flexible, but like the little bear's porridge to Goldilocks, "Just right." Standardization of processes also proves valuable as it creates the opportunity for software service providers to scale their operations without a one-to-one increase in labor (e.g. standardization may be a precursor to automating processes through the use of information technology – as is often done with processes for testing software).

4.4. The Principles as a System

Due to complementarities, the benefit from a lean system may be greater than the sum of the benefits from its parts (MacDuffie 1995; Shah and Ward 2007). For example, by standardizing practices, workers can set up valid experiments and then use their problem solving skills. The mutually reinforcing nature of lean practices helps build a culture where experimentation and useful failure are encouraged while mistakes are avoided (Sitkin 1992; Thomke 2003). These systemic benefits are relevant to software. Software developers are engaged in a debate as to whether the development of software is an art or a science (Beck and Boehm 2003). Commenting on this debate one executive at Wipro noted:

> IT is at the same place as the auto industry in 1910 – 1920. Today, each organization does their work uniquely; we are still in craft production. However, there is so much inefficiency that can be taken out and the market is too competitive for it not to be taken out. The entire outsourcing and globalization wave is proof that the transition is happening. Today you have to be able to deliver consistently across 10,000 minds. We don't know how to do this in software yet, but manufacturing does. A factory isn't about buying machines – it's about people.

In addition to the pressure for improving efficiency Wipro's customers were demanding more innovative business solutions. Drawing on lean production's proven ability to stretch out the productivity/innovation frontier (Adler et al. 1999), Wipro hoped to use the system as a whole to manage this tension.

5. The Lean Initiative at Wipro

We now examine two case studies of lean projects at Wipro. These cases illustrate how Wipro personnel changed their existing processes as they tried to map the ideas of lean production to their own context.

5.1. Tooling System

A global manufacturer provided over 1,000,000 production tools to suppliers worldwide and asked Wipro to develop an electronic system to track, audit, and reassign tools. The client used a three step process for IT development. Each step was completed by an outside party with oversight from the internal purchasing and IT organizations. The first step was the creation of a *business case*. Then a request for proposal (RFP) was issued for the second step, *plan and define*. The output from the second step was a set of documents which were used for the RFP for the final step, *construct and deploy*.

In this project a global systems integrator served as the prime contractor and Wipro completed the second and third steps. However, the client let six months pass between the completion of the plan-and-define step and the initiation of the construct-and-deploy phase. As a result of the delay, the original Wipro team had moved on to a new project and so Wipro assembled a new team to develop the system. Wipro staffed this project with eight people and expected it to take approximately six months.

The plan-and-define stage was completed under the assumption that the system would be standalone and web-based. However, prior to the initiation of construct-and-deploy the client decided to integrate the tooling system within its supplier management system, requiring the team to redo much of the earlier work. The supplier management system had been developed by another firm, so Wipro had to learn how the existing system functioned prior to development. Also, the new team found it difficult to get attention from the previous team since the old team was fully occupied with another project.

Prior to the start of the project the project manager (PM) was approached by his quality representative about using lean on this project. The PM was intrigued, having seen lean's beneficial impact on other projects within the account, and so he decided to try it out.

As part of the different approach to scheduling, the PM relied on the design structure matrix (DSM) technique. DSM highlights the various dependencies that exist in a project. In this case the PM

and technical lead spent one week building two DSMs – architectural and technical. The architectural DSM prioritized the modules in the system and then the technical DSM broke down the tasks within each module. As part of this process the PM also used a different approach for work allocation. For example, in the past if there had been five modules and five people then each person would get a module and twenty days to work on it. Using DSM the tasks were broken down into one or two day processes. The PM created three person teams and the team worked together to complete a module. In this case there was a front-end, middle and back-end and each team member had primary responsibility for one part. The team members rotated between the different parts for each module so that they gained additional skills. The PM used his tacit knowledge of project complexity and team member skills to assign work.

The team also used a visual control board (VCB) to highlight the status of the work in process. The PM placed an A4 sheet of paper in a central location with each team member's name and daily assignments for the week. At the end of each day the team member indicated what percentage of the work he had completed. The VCB not only provided a place for the PM to receive an overall status report (something that was not possible before) and a check for team member loading, but it also allowed him to identify potential problems sooner and then to provide targeted assistance as appropriate.

Inspired by the concepts of pull the team used an iterative approach to design. Since the "customer was not very sure what he wanted" the team found value in providing the client with working software (particularly user interfaces) as early as possible. The functional prototypes assisted in setting customer expectations. For example, with an early prototype the client wanted to enter "*" and have the software return all records immediately, "just like Google". The team was able to explain that they only had one server compared to Google's thousands of servers and so the cost of this functionality was too high given its value. Finally, the prototypes helped the team find bugs sooner, fix them more quickly, and avoid repeating the same mistakes. The PM estimated that 25% of the savings came from this last point.

Using root cause analysis led the team to implement two new approaches. First, requirement changes often created confusion with the client and additional work for Wipro. The team created a requirements clarification file to make sure that they fully understood all requirements and subsequent

changes prior to initiating work. They also developed a new internal coordination technique. While the offshore team was in charge of development the onsite team did much of the testing (permitting close to 24 hour per day development). Communicating errors between onsite and offshore was difficult as the two worked at different times and the offshore team often was unable to replicate an onsite error. To address this, the onsite team began using web video tools (WebEx) to capture errors. WebEx showed exactly the build/configuration and the actions that generated the error. Even though the teams were located thousands of miles apart they were able to go and see the problem themselves

Finally, the client had a rigid, formal delivery procedure with multiple parties and so required a month's notice to reschedule delivery. Therefore, on previous projects when teams had finished early they had been forced to wait on delivery thus giving back any potential savings. The team leveled tasks such as testing to spread effort over the entire time period, using fewer overall resources. This also enabled them to propose six revenue generating change requests to the client to fill additional time.

Overall the team saved 6.5% on effort and since this was a fixed price project, Wipro kept these savings. Coding productivity was 80+% higher than the company standard and total productivity was 10% better than the standard. The use of lean countermeasures helped the team cope with initial hurdles and substantial change to the project, but still deliver on time with added functionality and high quality. The PM noted, "for improvement the key is for the team to make lots of small changes. These things need to become part of the regular work where the team is constantly looking for things to improve."

5.2. Telecom Product

A global telecommunications firm, Telco, was a long time customer of Wipro. For several years Wipro developed and supported software for a particular product in Telco's offshore development center at Wipro. New releases came out every year and Wipro then supported and enhanced the product. Typically, a marketing, sales, and technology team at Wipro, with one person from Telco would develop new requirements for each release. Then these requirements went through a business readiness review at Telco. Wipro used these requirements to do all of the design before they wrote the code.

In August 2004, the project manager learned of the new lean initiative and inquired about taking part. The upcoming release had a time and materials contract and was due to last from August until October 2005, requiring 44 new features with a team of 15 people. Wipro would do the design, coding, and unit testing, an outside firm would verify the work, and then Telco would complete integration.

Projects for Telco typically used a waterfall project management approach (similar to stage gates, Royce 1970), but the PM decided to try an iterative model. She split the features into six phases and vetted the order with the customer. She then completed a DSM for the features in each phase. This highlighted unrecognized dependencies and also changed the priority order of the features. The PM noted, "the DSM takes 10 minutes and is more accurate as compared to spending a week and not getting it [the plan] exactly right."

After the first iteration, the client changed the project from Java to .Net (a change in architecture and language). The team was not experienced in .Net so they underwent two weeks of training. At this time the team added six inexperienced engineers as the client requested delivery five months earlier than planned. While Brooks' Law posits that due to coordination challenges, adding people to a late project makes it later (Brooks 1975), with the aid of lean ideas the team smoothly incorporated the new members.

The iterative approach was particularly useful. In the past the customer wanted to see demonstrations during design which involved generating dummy code (e.g. if the demo included a billing system, but the order entry system was not yet completed, then code to simulate order entry was needed). With the iterative approach the team was able to show actual output, which helped Telco see new possibilities for the system and expanded the scope of the project. With more iterations, engineers were able to try multiple experiments simultaneously and choose the best fit as they gained more information. For example, when the team needed to output information visually to a grid, the architect did not pick one way (as in the past), but rather used a three person tournament (where each person tried a different approach) to evaluate competing options. This expanded the search space for potential solutions.

Similar to the tooling project, the PM used a visual control board (VCB) on which work was divided into daily components. The project included many small tasks that previously were difficult to

track, but with the VCB the entire team could monitor progress. Automation of documentation, standards, and testing were all ideas generated from the team brainstorming about applying lean to their project. The team used Java documents to automate the creation of detailed design documents and put the client's coding standards into the integrated development environment to confirm that developers were using the right standards (i.e. code review) and immediately notify them if they were not (Stewart and Grout 2001). Additionally, the team instituted daily builds where the system was integrated for testing and also automated the unit test cases, so that code was automatically checked against pre-specified rules.

After final delivery of the development portion of the project, the team worked to improve the maintenance process. They completed a value stream map (VSM) of the bug fix process and discovered a lag in bug assignment. They redesigned their system so all bugs came into a queue and were then pulled by engineers when they completed their previous work. Using VSM the team realized that they had redundant reviews and so restructured their pathways, matching information and product flows.

Overall the project was a success as they delivered on time to the revised schedule. Quality for the project was better than the Wipro norm. The customer increased the scope so project revenue increased over twenty percent. Table 1 highlights examples from other lean projects at Wipro.

**********Insert Table 1 about here**********

We now examine how the practices Wipro implemented as part of the lean initiatives affected the delivery of customer value and improved learning and performance.

5.3. Problem Solving

The introduction of a lean production system helped increase experimentation and the devolution of responsibility to engineers. The former can be seen by the increase in the application of iterative methods at Wipro. In the past Wipro used a waterfall lifecycle project management approach where stages were completed sequentially (Royce 1970). In an iterative model, the aim is to build a working prototype rapidly, receive feedback from the customer, and then repeat the cycle. More functionality is built into each successive prototype. This allows project teams to experiment at the right level.

Iterative design is not new to software (Boehm 1985) or product development in general (Brown and Eisenhardt 1995). If iterative approaches have received such universal acclaim why has Wipro not adopted them previously? Boehm (1985) notes that, "If a project has … a high risk in budget and schedule predictability and control, then these risk considerations drive the spiral [iterative] model into an equivalence to the waterfall model." (p. 69) This accurately describes typical projects at Wipro.

The use of iterations addresses all four problem categories discussed earlier. Both cases highlight the variability in customer requests. Iterations help with customer learning, as they show a customer potential solutions. This helps codify the customer's tacit knowledge as well as educate the customer about the technological possibilities. An iterative model increases project-specific learning as engineers focus on the highest-value areas first and benefit from rapid feedback and low-cost experimentation (Thomke 1998). Iterations help with team learning as interdependencies are revealed sooner as testable output is generated early in the process. Team members can then use the feedback to make behavioral or architectural changes. Finally, iterations increase individual learning. In the past weeks or months might pass between when an engineer made a mistake and when it was found. With iterations the team finds errors more rapidly and problem and solution are kept closer together in time and space. As errors are identified sooner, rework is easier and individuals can avoid repeatedly making the same mistake.

Another part of the lean initiative was the increased use of periodic builds and code reviews. With a periodic build the system is integrated for testing, while a code review involves a check of an individual's code (either by a person or with software). Each is a type of jidoka tool. These two processes highlight team interdependencies and increase the opportunity for individual learning. They also help to identify errors sooner and to improve the psychological safety and reporting of errors as all team members are aware that others make errors and team leaders can provide extra assistance for those who need it (Edmondson 1999). Finally, periodic builds and code reviews improve the routinization of problem solving. Team members expect their work to be reviewed daily and to get immediate feedback. Learning cycles are triggered constantly. A project leader commented, "We traditionally had done a

batch and queue process. Now we do reviews close to one a day. Errors are reduced over the length of the project. Because of continuous review and continuous integration you catch the errors much sooner."

Finally, as mentioned in the telecom case, inspired by the idea of genchi genbutsu (go and see for yourself) a team developed a new problem solving and coordination technique for testing. By using technology (WebEx) to reach across the temporal and geographic separation (O'Leary and Cummings 2007) the teams were able to incorporate situated knowledge (Tyre and von Hippel 1997).

5.4. Coordination – Connections

Several practices that Wipro adopted within its lean initiative focus on improving coordination through connections. For example, visual control boards (VCB) provide a means for identifying direct connections. The VCB provides benefits for both the PM and team members. For the PM it offers information clarity by creating one place to receive a project status report. In the past the PM had to poll all team members to understand the project's status. The second benefit is that it forces the PM to split the work into small pieces. This assists with specifying the work to be completed and increasing the planning. Finally, it provides a simple, self-diagnostic test to assist with problem identification where the PM can see if she is under/over loading team members.

Initially team members were resistant to the VCB as they thought it was another monitoring tool and an additional "report" to fill out. However, over time most found it beneficial for multiple reasons. In the past they did not have access to the overall project plan, so the VCB provided a project summary. The VCB also helped them to learn about their own dependencies. So, if they were waiting on an input or had a question about an output they could go directly to the appropriate person. They could also see if the loading of work was equitable across the team and that they were not being unduly burdened. The VCB was also a self-diagnostic input for monitoring progress towards meeting their goals. Finally, the VCB provided an opportunity to identify problems so that individuals could get help sooner. One quality manager said, "It isn't the culture [at Wipro] to ask for help." Instead of waiting for major checkpoints, possible trouble spots were identified and workers could get assistance or training.

To help with architectural simplification and to help create direct connections and pathways Wipro introduced the design structure matrix (DSM). Using activities or functionalities of a project as inputs, the DSM outputs the project dependencies and suggests an ordering of tasks (see Smith and Eppinger 1997). Projects adopted DSM quickly because, "DSM offers a tool where someone inputs the data, pushes a button and then gets an answer." In other words, data are structured to define connections and paths. Describing DSM, a PM said:

> All of the improvement approaches we've used force a project manager to plan better and to monitor closer and both of these activities improve performance. With DSM, the PM has a tool to take the knowledge out of his head and put it onto paper.

A related tool, developed at Wipro is the system complexity estimator (SCE). It compares the actual architecture to the simplest possible architecture where each module would complete one task and not interact with others except through well-defined interfaces. The SCE measures deviation from the ideal and ranks modules based on complexity. Together DSM and SCE had a complementary impact at Wipro. By breaking work into small pieces, limiting dependencies, and allocating work to the appropriate personnel, the two tools helped to match product design and team structures (MacCormack et al. 2006).

5.5. Coordination – Pathways

Several new approaches with the lean initiative at Wipro addressed coordination through the restructuring of pathways. For example, teams used value stream mapping (VSM) to trace value adding processes and eliminate waste. Most teams used the VSM as a process flow diagram, which helped them to benefit from architectural simplification, information clarity and problem identification. They did not however, test the validity of the flow path against output parameters.

While every project was made to order since work did not start until a customer requested a project, inside a project work might follow a batch process. With the lean initiative projects tried to implement single piece flow to eliminate unnecessary work. Similarly, teams tried to apply heijunka. Heijunka is a coordination approach where work is leveled so production proceeds at a constant rate to meet customer demand. It addresses the issue that haste and slack time produce wasted resources and create defects. One PM described the value of the lean initiative in terms of coordination:

> Lean provided flow. It gave us interconnectedness. Testing thought development was the problem, development thought testing was the problem – now they could work together and see that each was the problem. … They ask, how do I fix myself? There is no finger pointing.

5.6. Standardization

The standardization of processes was an important part of Wipro's success both before and after the lean initiative. Prior to the lean implementation, Wipro had deployed process compliance approaches such as ISO-9000 and the CMMI. The lean initiative at Wipro pushed standardization even further through the use of standard error codes and tests, periodic builds and code reviews, DSM, SCE, and VSM. By making daily builds and code reviews a mandatory part of team member's processes, teams captured mistakes earlier and shared them with the team. Creating DSMs, SCEs, and VSMs required breaking work into smaller pieces offering the opportunity to standardize across production for projects with repetitive work. Standardization provided benefits in consistency of delivery, but even more importantly it offered an opportunity for learning. Once standard practices were identified then engineers could single out problems to address as well as areas for experimentation and improvement.

As shown, in their pursuit of a lean production system Wipro implemented many ideas that altered how the firm operated and learned. This changed how Wipro delivers customer value, helping to improve their flexibility and thus their performance in a dynamic environment (Anand and Ward 2004). The impact is seen in particular through problem solving, coordination and standardization. Wipro found that while individual tools were beneficial, combined together as a system, the value was amplified.

6. Data and Analysis

We next evaluate the lean initiative at Wipro empirically. We obtained detailed data on all lean and non-lean development projects completed between January 2005 and June 2007. After eliminating projects with missing data, our sample includes a total of 92 lean and 1,111 non-lean projects.[3] We examine development projects to provide a comparable sample across projects. Wipro executes many types of projects for its customers including maintenance and testing projects. These other types of projects are

[3] During this time a total of 772 lean projects were completed or underway. Of this total only 92 were development projects without missing data.

dependent upon the customer's context so it is difficult to compare them in a general way. Of the total sample, 65 lean and 662 non-lean development projects use kilolines of code (KLOC) as a unit of measurement and are included in the comparisons that use KLOC as a control for complexity.

6.1 Variables

Below we describe the outcome and control variables used in our analysis.

Schedule Deviation. We analyze schedule performance as a continuous variable, schedule deviation. Deviation can be positive or negative. Before a project begins both the schedule and effort are estimated and agreed to by the customer. During a project these estimates may be changed if the customer changes the scope of the project. The customer must formally sign off on any changes and Wipro has internal checks to prevent gaming of the system and to make sure that any changes are for legitimate business reasons. We use the revised estimates for both schedule and effort deviation as these most accurately reflect a project's final objectives. Schedule deviation is calculated using days as follows: $ScheduleDeviation = \frac{(DateDelivered - ScheduledDateDue)}{(ScheduledDateDue - StartDate)}$. We normalize for project length as we expect the potential of delays to increase with project length.

We also wish to examine the impact of the lean initiative on variation. To do this we calculate truncated deviation, coding all projects less than zero as zero and all projects greater than zero as above. This measure also allows us to analyze deviation which adversely impacts customers. If a project finishes with negative deviation then a customer receives at least the expected performance, however each day late is worse, even if by a decreasing amount, making this a useful measure of performance variation.

Effort Deviation. Similar to schedule performance we also examine effort deviation. It is calculated as follows: $EffortDeviation = \frac{(ActualEffort - EstimatedEffort)}{EstimatedEffort}$. We normalize the measure as we would expect misses in effort to be larger for larger projects. We also calculate truncated effort deviation to examine the impact of variation that negatively impacts customers.

Quality. To measure quality we use the number of defects in customer acceptance testing (CAT) divided by KLOC. CAT is the final stage of a project that occurs when a customer tests the code against

project requirements. We divide by KLOC to control for complexity. Not every project completes CAT and so in our data we are left with 50 lean projects and 474 non-lean projects.

Lean Project. We employ the company's definition for lean projects and use an indicator variable coded one for lean projects and zero otherwise. The company's broad definition – training followed by a good faith effort to apply lean thinking with at least one countermeasure – should make it more difficult to find an effect since it may include lean projects as well as those that are only superficially applying lean thinking. There are two ways by which it might be considered less conservative. First, there may have been a Hawthorne effect where the improvement was a result of increased attention rather than process changes (Roethlisberger and Dickson 1934). This seems less likely given Wipro's continuous focus on process improvement. Lean projects at Wipro did not receive significantly more attention or resources than non-lean projects. Also, as the number of projects increases the likelihood of a Hawthorne effect decreases. Second, it is possible that more able PMs choose to implement lean techniques and any superior performance derives from these better PMs. Managers at Wipro reported that this was not the case. To test the hypothesis we examined PM annual reviews. Each year employees were ranked on a four point scale. The average ranking for PMs on non-lean projects was 2.59 while for PMs on lean projects it was 2.71. This difference is not statistically significant and suggests similar PMs are running lean and non-lean projects, in terms of observable traits.

Project size and complexity. Since increasing project size and complexity may decrease operational performance we control for these factors using the estimated number of project hours, calendar days of the project and actual kilolines of manual code written for the project.

Type of contract. Wipro uses two typical contract structures: time and materials (T&M) and fixed price projects (FPP). In the first, a customer reimburses Wipro at a specified rate for the hours the project team works. In the second, a price is negotiated upfront and Wipro bears the risk of overages. We use a variable, FPP, coded one for a FPP contract and zero for a T&M contract.

Year. To control for learning and the environment, we include an indicator for project end year.

Strategic business unit. Wipro is divided into multiple Strategic Business Units (SBUs), which have varying competitive landscapes and strategic requirements, therefore we include indicator variables for each SBU. Table 2 provides summary statistics for our sample.

**********Insert Table 2 about here**********

6.2. Tests

To evaluate the performance of lean projects, we examine three samples. First, we compare lean to all non-lean projects. This does not account for differences in project traits, so we construct two matched samples. In the first, we identify control projects based on six factors: SBU, contract type, end year, total effort, duration, and KLOC. We match exactly on the first three traits and then select the matched project by minimizing the Euclidean distance using the formula: $\sqrt{\sum\left(\left(Effort_{lean}-Effort_i\right)^2+\left(Duration_{lean}-Duration_i\right)^2+\left(KLOC_{lean}-KLOC_i\right)^2\right)}$. To avoid issues with cross-sectional dependence we do not permit a project to be used as a match for more than one lean project. Our approach yields matches for all lean projects. We repeat the process, excluding KLOC to construct our second sample. This allows us to evaluate all of the lean development projects in our data, albeit with potentially less precise matches. Since all projects do not complete acceptance testing, not all matches have a quality value. We repeat the tests below for quality, restricting the pool so all matches have quality values and obtain similar results. Table 3 provides a sample breakdown.

**********Insert Table 3 about here**********

We use non-parametric tests, as *ex ante* we have no reason to assume normality. For each of our three samples we compare performance on schedule deviation, effort deviation, and quality. We also examine the impact of the lean initiative on variation, as measured by truncated schedule and effort deviation and quality. Table 4 provides summary results.

**********Insert Table 4 about here**********

In each of the tests, with respect to schedule deviation, we see that lean projects have lower average values (i.e. are more likely to finish early) than the comparison group. For the entire sample a

Wilcoxon rank-sum test rejects the null hypothesis that both samples are from the same distribution. For the matched samples we use a Wilcoxon matched-pairs signed-ranks test and it is significant for the sample without KLOC, but outside of conventional levels of significance for the KLOC matched sample. Examining Tables 2 and 3 reveals that the standard deviation of truncated schedule deviation is lower in the lean project samples than the non-lean project samples. Levene's test for the equality of variance (Levene 1960), rejects the null hypothesis that the two groups have equal variances in each sample.

When we examine effort deviation we see that the mean of lean projects is lower as compared to all non-lean projects and that the mean of the paired differences for the latter two samples is negative (implying lean projects perform better on this metric). Similar statistical tests as used in the schedule deviation case show that these differences are significant. From Tables 2 and 3 we see that the standard deviation for truncated effort deviation is lower in the lean versus the non-lean project samples. Levene's test statistic is significant for each comparison, rejecting the hypothesis that the variances are the same.

Finally, with respect to quality we see that the mean defect rate of lean projects is lower compared to all non-lean projects and the mean of the paired differences for the matched sample is lower. However, these differences are not statistically significant in either case. When we turn to the analysis of variance, the standard deviation is lower for the lean project samples as opposed to the non-lean project samples in Tables 2 and 3. Levene's test statistic is significant at the ten percent level providing partial support that the variance of the two groups is not the same.

6.3. Discussion

The data presented provide support that the lean initiative has positively impacted operational performance at Wipro. We find that lean projects have better schedule and effort performance than non-lean projects; however we do not see a significant difference in quality. Additionally, the variance of the lean project samples is lower than all three of the non-lean project samples.

While the lean initiative has had a positive impact on projects at Wipro the effect is not seen in all project outcome variables. Given lean production's positive impact on quality in manufacturing, there is a puzzling lack of an effect on quality in this setting. One possibility is that there will be an improvement

in quality, but that it will take more time. For example, it may be necessary to build in learning processes after customer acceptance testing and that these processes take time to create. Second, the impact of lean may not be seen in traditional measures of quality. As the earlier comments suggested, quality as measured by defects may not be the key competitive performance variable going forward. Quality can be measured many ways and so the impact of a lean system in software services may be seen in other, currently unmeasured quantities, such as customer value, i.e. performance quality (Garvin 1987).

Wipro personnel suggested two additional factors that may heighten the revealed performance impact of the lean initiative over time . First, many early projects adopted lean principles after the project fell behind and they were looking for a way to get back on track. We discovered a number of examples like this. Software engineering suggests that rescuing a troubled project is difficult (Brooks 1975), but in our analysis we only evaluate the end state so we would miss if lean principles helped "save" a project. A second explanation is that in an effort to show quantifiable results quickly, many early lean projects touched only one part of the overall project. That part of the project (e.g. coding and unit testing) may have experienced savings, but when translated to the entire project the gain was much smaller. Also, some projects were not prepared for gains in one part of the project and so gave them back in a later part.

We reviewed the full lean project closure reports for all development projects in one business unit (38 projects). In this sample, the lean engagement encompassed on average 58% of the total project although this varied from a low of 14% to a high of 100%. This suggests a continued opportunity to increase the size of the lean engagement within projects.

In addition to the company-specific contingencies discussed, it is valuable to consider systemic reasons as to why lean ideas from manufacturing might *not* be applicable in software services. First, it is possible that there is greater uncertainty in software than in manufacturing and that this invalidates the underlying thinking of lean. While most people think of the shop floor when they hear lean, Toyota uses the same guiding principles in its product development efforts and TPS has been applied to arenas with considerable uncertainty such as aircraft engine design (Bowen and Purrington 2004). While uncertainty may alter some of the practices of lean it seems unlikely that it will completely nullify the approach.

Second, despite the significant efforts at standardizing and codifying knowledge within TPS (Ferdows 2006), much of the knowledge about the system is tacit and resides in workers' actions and heads, not on paper (Nonaka 1994; Orlikowski 2002). A result of this is the lack of a consistent definition of lean (Shah and Ward 2007). Many services industries face turnover concerns, for example the Indian software services industry had double digit average annual personnel turnover, so it may be that there is insufficient consistency in personnel in most service companies to implement a lean model successfully.

Next, implementing lean production in services presents different challenges from manufacturing since the customer is involved in production (Frei 2006). While Wipro's processes are CMMI Level 5 (i.e. standardized and disciplined) many of its customer's processes are not. Talking about his CMMI Level 2 client, a project manager noted that "Some amount of tailoring our work to the specifics of the customer's processes is necessary – and this certainly complicates things." This is analogous to Toyota's experience with its suppliers. To implement its operating model effectively Toyota invested significant resources in training its suppliers (Fujimoto 1999). Toyota has had to repeat this process as they have entered new countries (Mishina 1992). It is likely that for the lean initiative at Wipro to have lasting impact they will have to both involve and train customers. This is complicated by the potential asymmetrical power relationship between a firm and its customers as opposed to a firm and its suppliers.

Finally, it is possible that these (and other) issues are surmountable, but only with time. If the greatest value from lean production comes when complementary practices are implemented as a system then this may take some time as the company slowly explores to find the right contextually dependent set of practices that will eventually make up the "Wipro Production System." This time frame may prove beneficial if it makes it more difficult for competitors to emulate Wipro (C.f. Dierickx and Cool 1989).

7. Conclusion

During a meeting of the quality leaders at Wipro one attendee asked about the lean initiative, "Most of these things are in software engineering already. Are we just repackaging?" In an interview a PM said, "Lean is good. It just uses common sense and it actually delivers value." Despite the simplicity of the lean principles and ideas and the openness of Toyota it has been difficult for many companies to put lean

production systems in place. TPS has proven quite robust and sustainable as a source of competitive advantage for Toyota (Taylor 2007). In this paper we see that the implementation of a lean production system in services changes how an organization learns through problem solving, coordination through connections and pathways, and standardization. In its attempt to implement a lean production system we see that core processes were altered and that this resulted in improved operational performance.

These assessments suggest the applicability of manufacturing-based principles to a fast-moving, high technology, service industry. Our observations of the details of the implementation will hopefully provide the beginnings of a roadmap for other service industries seeking to apply the same ideas (C.f. Boyer et al. 2005). Such details are the most important (and most often under-emphasized) part of any lean initiative, and far outstrip the import of a strategic mandate that 'we are doing lean'.

Like any study, ours has limitations, and one should be cautious in applying its results. Our study examines implementation of lean production in services, but it is the experience of one company. It is possible that our observations will not generalize to other settings. Also, while the interim results at Wipro are promising, the implementation has far to go to deliver fully on its promise. We think that the detail and lack of recall bias that the real-time nature of our study permits more than compensates for this limitation, although the concern is real. Finally, there is the question of whether what Wipro is doing is "lean." Our study does not rely on the epistemological concern of whether Wipro's approach was truly lean. Since there is not an accepted definition of lean in services in general, or software in particular, we rely on the fact that Wipro was consciously *trying* to create a lean system for software services. Their ideas were motivated by lean thinking and so we are able to learn from their attempted mapping.

While the implementation of its lean production system is far from complete, it may offer Wipro a way to manage uncertain and complex projects through a high assurance, iterative model. The introduction of an innovative process for managing software projects creates new opportunities for the firm to learn (Pisano 1996). Having previously followed a public, external approach to process management that left the company open to imitation, the switch to lean techniques may create the opportunity for an ongoing operations-based competitive advantage.

References

Abrahamson, E. and G. Fairchild (1999). "Management fashion: Lifecycles, triggers, and collective learning processes." *Administrative Science Quarterly* **44**(4): 708-740.

Adler, P. S. (2005). "The evolving object of software development." *Organization* **12**(3): 401-435.

Adler, P. S., B. Goldoftas and D. I. Levine (1999). "Flexibility versus efficiency? A case study of model changeovers in the Toyota Production System." *Organization Science* **10**(1): 43-68.

Altshuller, G., L. Shulyak and S. Rodman (1999). *The Innovation Algorithm: TRIZ*. Worcester, MA, Technical Innovation Center.

Anand, G. and P. T. Ward (2004). "Fit, flexibility and performance in manufacturing: Coping with dynamic environments." *Production and Operations Management* **13**(4): 369-385.

Beck, K. and B. Boehm (2003). "Agility through discipline: A debate." *IEEE Computer* **36**(6): 44-46.

Boehm, B. (1981). *Software Engineering Economics*. New York, Prentice Hall.

Boehm, B. (1985). "A spiral model of software development and enhancement." *Proceedings of an International Workshop on Software Process and Software Environments*.

Bohn, R. E. (1995). "Noise and learning in semiconductor manufacturing." *Management Science* **41**(1): 31-42.

Bowen, H. K. and C. Purrington (2004). "Pratt & Whitney: Engineering Standard Work." Harvard Business School Case # 604084.

Boyer, K. K., M. Swink and E. D. Rosenzweig (2005). "Operations strategy research in the POMS Journal." *Production and Operations Management* **14**(4): 442-449.

Brooks, F. (1975). *The Mythical Man-Month: Essays on Software Engineering*. New York, Addison-Wesley.

Brown, S. L. and K. M. Eisenhardt (1995). "Product development: Past research, present findings, and future directions." *Academy of Management Review* **20**(2): 343-378.

Clark, K. B. (1988). Managing technology in international competition: The case of product development in response to foreign entry. in *International Competitiveness*. Eds. M. Spence and H. A. Hazard, Ballinger**:** 27-74.

Clark, K. B. and T. Fujimoto (1991). *Product Development Performance*. Boston, Harvard Business School Press.

Cusumano, M. A. and K. Nobeoka (1998). *Thinking Beyond Lean: How Multi-Project Management is Transforming Product Development at Toyota and Other Companies*. New York, Free Press.

de Treville, S. and J. Antonakis (2006). "Could lean production job design be intrinsically motivating? Contextual, configurational, and levels-of-analysis issues." *Journal of Operations Management* **24**(2): 99-123.

Dierickx, I. and K. Cool (1989). "Asset stock accumulation and the sustainability of competitive advantage." *Management Science* **35**(12): 1504-1511.

Edmondson, A. (1999). "Psychological safety and learning behavior in work teams." *Administrative Science Quarterly* **44**(2): 350-383.

Eisenhardt, K. M. (1989). "Building theories from case study research." *Academy of Management Review* **14**(4): 532-550.

Ferdows, K. (2006). "Transfer of changing production know-how." *Production and Operations Management* **15**(1): 1-9.

Frei, F. X. (2006). "Breaking the trade-off between efficiency and service." *Harvard Business Review* **84**(11): 92-101.

Fujimoto, T. (1999). *The Evolution of a Manufacturing System at Toyota*. New York, Oxford University Press.

Garvin, D. A. (1987). "Competing on the Eight Dimensions of Quality." *Harvard Business Review* **65**(6): 101-110.

Hayes, R. H. (1981). "Why Japanese factories work." *Harvard Business Review* **59**(4): 57-66.

Hayes, R. H. and G. P. Pisano (1996). "Manufacturing strategy: At the intersection of two paradigm shifts." *Production and Operations Management* **5**(1): 25-41.

Hill, T. (1993). *Manufacturing Strategy: Text and Cases*. Blue Ridge, Ill., Irwin.

Hino, S. (2006). *Inside the mind of Toyota*. New York, Productivity Press.

Hopp, W. J. and M. L. Spearman (2004). "To pull or not to pull: What is the question?" *Manufacturing Service Operations Management* **6**(2): 133-148.

Krafcik, J. F. (1988). "Triumph of the lean production system." *Sloan Management Review* **30**(1): 41-52.

Kraut, R. E. and L. A. Streeter (1995). "Coordination in software development." *Communications of the ACM* **38**(3): 69-81.

Levene, H. (1960). Robust tests for equality of variances. in *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling* Eds. I. Olkin. Palo Alto, CA, Stanford University Press**:** 278-292.

Li, S., S. Subba Rao, T. S. Ragu-Nathan and B. Ragu-Nathan (2005). "Development and validation of a measurement instrument for studying supply chain management practices." *Journal of Operations Management* **23**(6): 618-641.

Liker, J. K. (2004). *Toyota Way*. New York, McGraw-Hill.

MacCormack, A., J. Rusnak and C. Y. Baldwin (2006). "Exploring the structure of complex software designs: An empirical study of open source and proprietary code." *Management Science* **52**(7): 1015-1030.

MacCormack, A. and R. Verganti (2003). "Managing the sources of uncertainty: Matching process and context in software development." *Journal of Product Innovation Management* **20**(3): 217-232.

MacDuffie, J. P. (1995). "Human resource bundles and manufacturing performance." *Industrial and Labor Relations Review* **48**(2): 197-221.

MacDuffie, J. P. (1997). "The road to "Root Cause": Shop-floor problem-solving at three auto assembly plants." *Management Science* **43**(4): 479-502.

MacDuffie, J. P., K. Sethuraman and M. L. Fisher (1996). "Product variety and manufacturing performance: Evidence from the international automotive assembly plant study." *Management Science* **42**(3): 350-369.
McCarthy, J. C. (2004). "Low-cost global delivery model showdown." *Forrester Research, Inc.* Retrieved July 16, 2007.
Middleton, P. and J. Sutton (2005). *Lean Software Strategies*. New York, Productivity Press.
Mishina, K. (1992). "Toyota Motor Manufacturing, U.S.A., Inc." Harvard Business School Case # 693-019.
Monden, Y. (1983). *Toyota Production System: Practical Approach to Management*. Norcross, GA, Industrial Engineering and Management Press.
Narasimhan, R., M. Swink and S. W. Kim (2006). "Disentangling leanness and agility: An empirical investigation." *Journal of Operations Management* **24**(5): 440-457.
NASSCOM. (2005). "NASSCOM–Hewitt total rewards study." Retrieved April 20, 2006, from http://www.nasscom.in/custompages/Hewittstudy/.
Nelson, R. R. and S. G. Winter (1982). *An Evolutionary Theory of Economic Change*. Cambridge, MA, Belknap Press.
Nonaka, I. (1994). "A dynamic theory of organizational knowledge creation." *Organization Science* **5**(1): 14-37.
O'Leary, M. B. and J. N. Cummings (2007). "The spatial, temporal, and configurational characteristics of geographic dispersion in teams." *MIS Quarterly* **31**(3): 433-452.
Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Cambridge, Mass., Productivity Press.
Orlikowski, W. J. (2002). "Knowing in practice: Enacting a collective capability in distributed organizing." *Organization Science* **13**(3): 249-273.
Pisano, G. P. (1996). "Learning-before-doing in the development of new process technology." *Research Policy* **25**(7): 1097-1119.
Poppendieck, M. and T. Poppendieck. (2003). *Lean Software Development: An Agile Toolkit*. Boston, Addison Wesley.
Roethlisberger, F. J. and W. J. Dickson (1934). *Management and the Worker*. Boston, Harvard University.
Royce, W. (1970). "Managing the development of large software systems." *Proceedings, IEEE Wescon*.
Shah, R. and P. T. Ward (2003). "Lean manufacturing: Context, practice bundles, and performance." *Journal of Operations Management* **21**(2): 129-149.
Shah, R. and P. T. Ward (2007). "Defining and developing measures of lean production." *Journal of Operations Management* **25**(4): 785-805.
Sitkin, S. B. (1992). Learning through failure: The strategy of small losses. in *Research in Organizational Behavior*. Eds. L. L. Cummings and B. M. Staw. Greenwich, CT, JAI Press. **14**.
Smith, R. P. and S. D. Eppinger (1997). "Identifying controlling features of engineering design iteration." *Management Science* **43**(3): 276-293.
Sousa, R. and C. A. Voss (2001). "Quality management: Universal or context dependent?" *Production and Operations Management* **10**(4): 383-404.
Spear, S. and H. K. Bowen (1999). "Decoding the DNA of the Toyota Production System." *Harvard Business Review* **77**(5): 97-106.
Spear, S. J. (1999). *The Toyota Production System : An Example of Managing Complex Social/Teaching Systems*.
Stewart, D. M. and J. R. Grout (2001). "The human side of mistake-proofing." *Production and Operations Management* **10**(4): 440-459.
Strauss, A. L. and J. M. Corbin (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Newbury Park, CA, Sage Publications.
Taylor, A., III (2007). "America's best car company." *Fortune* **155**(5): 98-106.
Thomke, S. H. (1998). "Managing experimentation in the design of new products." *Management Science* **44**(6): 743-762.
Thomke, S. H. (2003). *Experimentation Matters*. Boston, Harvard Business School Press.
Tucker, A. L. (2004). "The impact of operational failures on hospital nurses and their patients." *Journal of Operations Management* **22**(2): 151.
Tyre, M. J. and E. von Hippel (1997). "The situated nature of adaptive learning in organizations." *Organization Science* **8**(1): 71-83.
Ulrich, K. (1995). "The role of product architecture in the manufacturing firm." *Research Policy* **24**(3): 419-440.
von Hippel, E. (1990). "Task partitioning: An innovation process variable." *Research Policy* **19**(5): 407-418.
Ward, A., J. K. Liker, J. J. Cristiano and D. K. Sobek, II (1995). "The second Toyota paradox: How delaying decisions can make better cars faster." *Sloan Management Review* **36**(3): 43-51.
Winter, S. G. and G. Szulanski (2001). "Replication as strategy." *Organization Science* **12**(6): 730-743.
Womack, J. P. and D. T. Jones (1994). "From lean production to the lean enterprise." *Harvard Business Review* **72**(2): 93-103.
Womack, J. P. and D. T. Jones (1996). *Lean Thinking*. New York, Simon & Schuster.
Womack, J. P. and D. T. Jones (2005). "Lean consumption." *Harvard Business Review* **83**(3): 59-70.
Womack, J. P., D. T. Jones and D. Roos (1990). *The Machine That Changed the World*. New York, Rawson Associates.
Yin, R. K. (2003). *Case Study Research: Design and Methods*. Thousand Oaks, Calif., Sage Publications.
Zollo, M. and S. G. Winter (2002). "Deliberate learning and the evolution of dynamic capabilities." *Organization Science* **13**(3): 339 - 351.

Figures & Tables

Figure 1. Indian Software Services' Industry Evolution

Table 1. Examples of Countermeasures Deployed at Wipro as Part of the Lean Initiative

Problem Solving	*Use of iterations:* A financial services team had previously defined the team in three pieces doing different layers (e.g. web, middleware, database). Inspired by single piece flow they shifted to a feature based model with iterations of multiple features, improving integration within the team. By delivering high value items first the team was able to work around problems as they arose. The customer appreciated that he could provide in-process direction.	*Periodic Builds:* A project team for a high technology client was having delays with testing and configuration management. To address this they switched to daily builds and more frequent testing. This not only helped them to identify errors sooner, but they also realized that they could shift to two people testing for four months instead of four people over two months. This eliminated a testing equipment bottleneck.	*Periodic Code Reviews:* A project team working for a Japanese client was developing country customized applications for the firm's financial systems. They created a multi-tiered review system. They increased peer reviews by team leaders from every 5-6 days to every 2-3 days. They chose not to load experienced people fully, but leave 1-2 hours per day for reviews. Also, they added a self-review with a checklist.
Coordination – Connections	*Visual Control Boards (VCB):* A project team working for a manufacturing client created a visual control board which included all of the team members and the tasks for the week. At the bottom of the sheet, the PM put independent projects that people could take if they finished all of their work. A manager noted, "With the visual control board the team was able to understand dependencies – they could feel dependencies and their importance. We got peer competition out of peer pressure. Guys saw who was struggling and went and helped. We learned how to load the team."	*Design Structure Matrix (DSM):* A large client wanted a proof of concept for porting a mobile sales application to Windows CE from Palm OS. The project was scheduled for four weeks, but after completion of the project plan the client asked for delivery in three weeks. The PM asked the Productivity Office for help and the first step was to create a DSM. The 100x100 matrix of functionality showed that the PM had missed several dependencies in the plan. It also helped to identify common components. With the help of the lean ideas the project was finished in two weeks and the client approved the overall project.	*System Complexity Estimator (SCE):* An example of SCE occurred on a project where the PM used DSM to create the project plan, but wanted to see if the team composition was correct. He ran SCE to cluster the modules as simple, medium, and complex and then ranked team members' skills as low, medium, and high. This revealed that he had too few highly skilled team members for the complex modules. It identified a need for increased training and mentoring so the medium skilled team members would be able to complete the work. The PM also reordered the work to maximize the use of his highly skilled team members.
Coordination – Pathways	*Value Stream Mapping (VSM):* From the VSM one team found that four people were using the same test printer resulting in wasted time from waiting and changeovers. The printer was on another floor so if someone found an error he had to go down stairs make the change and print again. The team scheduled slots for the printer and set up an adjacent computer for testing.	*Single Piece Flow:* A team was upgrading a client's website and the prior version was built on 680 Java Server Pages (JSPs). Web pages could call one or several JSPs. Previously one team would work on the web pages while another converted the JSPs. On this project each web page moved through production in a single piece flow so JSPs were not converted until they were called. At the end of the process the team found that 200 of the JSPs were not called on the new site and did not need to be converted.	*Heijunka:* A team used heijunka after they completed their first lean project early, but wasted the savings waiting for other providers. This waste identified that certain people within the pathway were unnecessary for this project since the work was finished too quickly. On the next project the team used their full time allocation, but did the work with fewer people resulting in savings for Wipro.
Standardization	*Standardized Error Codes:* A project had "lots of debates for error classification." The PM assembled the team leads and created a system of standardized error codes. They then built an automated jidoka tool for pre-review to verify that developers were using the rules. This led to zero client errors where on the previous smaller project, they had twenty errors.	*5S:* A testing project for a technology customer used 5S to organize their lab resulting in improved productivity. As part of the project, the client sent them equipment for testing. Due to government regulations it was inefficient to send the printers back and Wipro could not just throw the equipment away. The 5S process revealed substantial wasted effort in sorting through the equipment so the team came up with standard approaches for the lab.	*DSM & SCE:* A team was building and converting multiple forms across countries for a client and decided to undertake lean principles. After completing the DSM, SCE, and VSM, they realized that the process for each form was very similar and they could standardize their approach. This approach led to fewer defects, less rework and improved productivity.

Table 2. Summary Statistics of Dependent, Independent and Control Variables of Interest

Variable	n	Mean	σ	Min	Max
Lean Projects					
Effort (hours)	92	13,470	14,564	1,586	83,127
KLOC	65	118.9	138.3	7.0	848.9
FPP	92	0.72	0.45	0.00	1.00
Schedule Deviation (%)	92	-4.86	14.85	-84.56	44.26
Truncated Schedule Deviation	92	1.09	5.14	0.00	44.26
Effort Deviation (%)	92	-9.40	9.68	-50.08	6.97
Truncated Effort Deviation	92	0.22	0.94	0.00	6.97
Quality	50	0.13	0.23	0.00	0.93
Non-Lean Projects					
Effort (hours)	1111	8,362	12,919	67	268,253
KLOC	662	68.2	213.1	0.3	4,207.1
FPP	1111	0.63	0.48	0.00	1.00
Schedule Deviation (%)	1111	-0.80	19.72	-81.12	250.28
Truncated Schedule Deviation	1111	3.63	14.04	0.00	250.28
Effort Deviation (%)	1111	-4.81	25.85	-95.06	415.64
Truncated Effort Deviation	1111	3.77	20.97	0.00	415.64
Quality	474	0.37	1.63	0.00	21.53

Variable	n	1	2	3	4	5	6	7	8
1. Effort (hours)	1203								
2. KLOC	727	**0.46**							
3. FPP	1203	**-0.10**	-0.05						
4. Schedule Deviation	1203	**0.07**	0.05	0.00					
5. Truncated Schedule Deviation	1203	0.04	0.03	0.06	**0.76**				
6. Effort Deviation	1203	**0.08**	0.07	-0.05	**0.27**	**0.24**			
7. Truncated Effort Deviation	1203	**0.06**	0.06	0.01	**0.19**	**0.24**	**0.87**		
8. Quality	524	0.03	-0.01	0.02	0.00	-0.02	0.05	0.02	
9. Lean Project	1203	**0.10**	0.07	0.05	-0.06	-0.05	-0.05	-0.05	-0.05

Note. Bold denotes significance of less than 5%.

Table 3. Breakdown of Dependent Variables between Lean and Matched Samples

		Lean Projects		Control Group	
Dependent Variable	**n**	**Mean**	**σ**	**Mean**	**σ**
Matched Sample Excluding KLOC					
Schedule Deviation (%)	92	-4.86	14.85	-0.25	14.87
Truncated Schedule Deviation	92	1.09	5.14	3.16	9.22
Effort Deviation (%)	92	-9.40	9.68	-1.95	28.55
Truncated Effort Deviation	92	-1.95	28.55	5.55	25.50
Matched Sample Including KLOC					
Schedule Deviation (%)	65	-3.13	14.63	-0.38	14.96
Truncated Schedule Deviation	65	1.51	6.08	2.85	8.55
Effort Deviation (%)	65	-9.43	9.54	-4.55	10.03
Truncated Effort Deviation	65	0.15	0.58	1.56	5.68
Quality	36	0.15	0.25	0.60	3.00

Table 4. Summary Results for Tests of Lean Projects versus Three Comparison Groups

	Lean Projects Tested Against:		
Dependent Variable	**Entire Dataset**	**Match no KLOC**	**Match with KLOC**
Schedule Deviation			
Mean of the paired differences[1]	N/A	-4.606	-2.758
Wilcoxon test Z statistic[2]	2.185**	-2.132**	-0.567
Truncated Schedule Deviation			
Levene's test statistic for the equality of variances[3]	9.735***	11.314***	4.648**
Effort Deviation			
Mean of the paired differences[1]	N/A	-7.445	-4.875
Wilcoxon test Z statistic[2]	3.334***	-2.660***	-2.297**
Truncated Effort Deviation			
Levene's test statistic for the equality of variances[3]	7.849***	14.449***	12.224***
Quality[4]			
Mean of the paired differences[1]	N/A	N/A	-0.445
Wilcoxon test Z statistic[2]	-0.140	N/A	-0.701
Levene's test statistic for the equality of variances	3.120*	N/A	3.145*

Note: *, ** and *** denote signficance at the 10%, 5% and 1% levels, respectively, for two-tailed tests. The sample sizes are as follows: the entire dataset column compares the 92 lean projects to all 1,111 non-lean projects (50 and 474 projects for quality); the match no KLOC column compares 92 matched lean and non-lean projects; the match with KLOC column compares 65 lean and non-lean projects (50 for quality).

(1) As the entire dataset comparison is not a matched set, there is no paired difference therefore, that value is not available.

(2) For the entire dataset comparison, since the data is unmatched we run a Wilcoxon rank-sum test, while for the latter two comparisons which are matched pairs we run a Wilcoxon matched-pairs signed-ranks test.

(3) We calculate Levene's test statistic for equality of variance between each of the lean and non-lean project samples.

(4) Since our quality measure is scaled by kilolines of code the measure is not used in the sample without KLOC